Development of Fast One-Dimensional Model for Prediction of Coupled Electrochemical-Thermal Behavior of Lithium-Ion Batteries

Undergraduate Honors Thesis

By

Jiheng Lu

Department of Mechanical Engineering
The Ohio State University
Columbus, OH, 43201, USA

Advisor:

Dr. Sandip Mazumder

Department of Mechanical Engineering
The Ohio State University
Columbus, OH, 43201, USA

March 2013

ABSTRACT

Spatially and temporally resolved one-dimensional transient models for the prediction of performance of lithium-ion batteries have been prevalent in the literature for over a decade. It is generally believed that such models that take into account the detailed mass transfer and electrochemistry within the battery are unsuitable for real-time control of batteries. As a result, several attempts have also been made to develop reduced-order approximate models that are suitable for real-time control. While these reduced models are efficient, they fail in non-linear regimes of operation. In this thesis, it will be shown that full spatial and temporal resolution of the battery with the inclusion of detailed transport phenomena and electrochemistry is possible with faster-than-real-time computing times provided appropriate numerical techniques are employed. The model presented here employs the same governing conservation equations of mass, energy, and charge as employed in previous studies. Only, the numerical procedure and solution algorithms are different. These are presented in some detail. The model was first successfully validated against experimental data for both charge and discharge processes in a Li_xC_6— $LiMn_2O_4$ battery. Finally, it was demonstrated for an arbitrary load typical of a hybrid electric vehicle drive cycle. The model was able to predict the cell voltage of a 15-minute drive cycle in less than 9 minutes of compute time on a laptop with a 2.3 GHz Intel i7 processor.

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Dr. Sandip Mazumder for the opportunity he granted me to work with him and the continuous support he provided. During the whole research process, good things and bad things happened. I am really very grateful that Dr. Mazumder trusted me and provided solid support to me. None of this work could have been accomplished without his patient guidance and help.

My appreciation also goes to Mechanical Engineering Honors Program. This research was supported by an undergraduate honors research fellowship awarded by the College of Engineering, the Ohio State University. Thanks to their generous support, the research became successful.

VITA

Jiheng Lu was born January 18th, 1990 at Shanghai , China. He grew up in Shanghai with his parents Xuefeng Lu and Yongfang Zhou. He is the only child in his family. After graduating from First Tonji High School, in September, 2009, Jiheng left Shanghai and came to the United States alone. He started to study in the Ohio State University and is majoring in Mechanical Engineering. He attained a wide breadth of knowledge through course work and research. He founded and led ACT4U (Assistant and Consultant Team For You) to help international students to get involved in campus activities, encourage and provide support for them to participate in on-campus part-jobs and internships. After graduation, Jiheng hopes to further deepen his knowledge by seeking a master's degree.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

LETTERS

$A_{collector}$	Collector plate surface area (m^2)
a_s	Active surface area to volume ratio of the electrode (m^{-1})
C_e	Molar concentration of Li^+ ions in the electrolyte (mol/m^3)
c_p	Average specific heat capacity over all phases (J/kg/K)
$C_{s,avg}$	Average solid-phase concentration of Li^+ ions within the active particle (mol/m^3)
$C_{s,max}$	Maximum concentration of Li^+ ions in the solid phase (mol/m^3)
$C_{s,surface}$	Concentration of Li^+ ions at the surface of the active particle (i.e., at the solid-electrolyte interface) (mol/m^3)
D_e	Free-stream diffusion coefficient of lithium-ions in the electrolyte (m^2/s)
D_e^{eff}	Effective diffusion coefficient of Li+ ions within the porous (m^2/s)
D_s	Diffusion coefficient of lithium-ions within the active particles (m^2/s)
f	Mean molar activity coefficient
F	Faraday constant (96487 C/mol)
i_0	Exchange current density

I_{app}	Applied current (drawn current) or the load (A)
j_{Li}	Transfer current
k^{eff}	Average thermal conductivity over all phases (W/m/K)
$\dot{q}$	Volumetric heat generation rate within the battery (W)
R	Universal gas constant (8.314 J/mol/K)
$R_{contact}$	Contact resistance between the electrodes and the current collectors (Ω)
R_{SEI}	Resistance due to irreversible film formation at solid-electrolyte interface (Ω)
t_+	Transference number
T	Absolute temperature (K)
U	Open circuit potential (or equilibrium potential) (V)
x, y	Stoichiometry in the negative and positive electrodes respectively

GREEK

α_a, α_c	Kinetic constants that depend on the chemical composition of the electrodes and the electrochemical reactions that occur on the active particles
ε_e	Volume fraction of the electrolyte containing part of the porous electrodes and separator (i.e. porosity)
ε_s	Volume fraction of the active particles in the electrode
η	Overpotential (V)
$\kappa^{eff}, \kappa_D^{eff}$	Effective ionic and diffusional conductivities (W/m/K)
ρ	Average density over all phases (kg/m^3)
σ^{eff}	Actual electrical conductivity of particles in electrodes (s/m)
ϕ_e, ϕ_s	Electrolyte (ionic) and solid (electronic) phase potentials (V)

Chapter 1

INTRODUCTION

1.1. Background

Lithium-ion (Li-ion) batteries are one of the most widely used energy storage systems nowadays because of their advantages including high energy storage density, high open circuit voltage output, no memory effect, among others. Li-ion batteries are commonly utilized for consumer electronic devices, and are the most popular type of rechargeable batteries for portable electronics. They can be found in daily-use cell phones (Fig. 1), tablets, laptops, and other portable devices. For the Li-ion batteries used in these devices, the technology is relatively mature, and the performance and behavior of batteries are relatively well understood by researchers.

In the past decade or so, electrical vehicles and hybrid electrical vehicles (HEV) have come to the forefront (Fig. 2). Because of its advantages of being lighter and smaller than nickel-metal hydride batteries, Li-ion batteries are becoming the primary mean of energy storage in this new application area. The reason Li-ion batteries was

Figure 1 - Lithium-ion batteries are the most popular energy storage technology for commercial uses. Figure shows its use in a cell phone [Hughes, 2012].

Figure 2 - The Lithium-ion battery pack under testing for new models of Prius, one of the most popular hybrids electrical vehicles in commercial world [Thomas, 2007].

not initially chosen as the energy storage system for hybrid cars is that it was once considered too unsafe for use in vehicles. After further research and improvement, however, nowadays Li-ion batteries have been deemed much safer and "greener" than before as well as other types of batteries [Thomas, 2007].

Currently, Lithium-ion batteries are more expensive than using internal combustion engines directly in cars. However, since the technology is "green," it is still considered a worthwhile technology to pursue and improve, so that the cost can be reduced further while at the same time, the performance of these batteries is improved.

From the scientific standpoint, there are two aspects of Li-ion batteries that are currently not well understood. First, when a Li-ion battery is repeatedly charged and discharged, the cell voltage decreases over time. While, in general, this is attributed to certain irreversible reactions that occur within the battery, the exact cause of this "aging" phenomenon is not well understood. Secondly, it has been observed that when a battery is charged / discharged either too rapidly or to extremely low capacity, it ages more quickly or is permanently damaged.

Lithium-ion batteries can be of various types, including Lithium-cobalt-oxide (LCO), Lithium-nickel-manganese-cobalt-oxide (NMC), Lithium-nickel-cobalt-aluminum-oxide (NCA), Lithium-iron-phosphate (LFP), Lithium-manganese-oxide (LMO), and Lithium-titanate (LTO). These types differ due to the anode and cathode materials utilized. New cathode and anode materials continue to be discovered, along with new electrolytes [HEV, 2013].

Across different types of Lithium-ion batteries, chemistry, performance, cost, and safety characteristics vary a lot. Handheld electronics mostly use Lithium-cobalt-oxide (LCO), which offers relatively high power density, but has safety concerns; Lithium-iron-phosphate (LFP), Lithium-manganese-oxide (LMO), and Lithium-nickel-manganese-cobalt-oxide (NMC) offer relatively lower energy density, but longer lives and more reliable safety performance. These types are being widely used for electric tools, medical equipment, etc. For automotive application, LMO is the leading contender. Other types, including Lithium-nickel-cobalt -aluminum-oxide (NCA) and Lithium-titanate (LTO), are less used, but these chemistries are still very valuable and such batteries can be specially designed to meet specific requirements [HEV, 2013].

1.2. Literature Review and Relevant Works

To the best of the author's knowledge, the earliest model for the simulation of Lithium-ion battery performance was developed by Newman and his co-workers [Doyle, 1993]. Later, this pioneering work was improved and extended by White and his co-workers [Ramadass, 2003]. Until the late 1990s, the researchers still mainly focused on one-dimensional (1D) simulations, even though the governing equations proposed, included general conservation equations of mass, charge, and energy, and are valid for any geometry. Although somewhat simplistic, these 1D models have found widespread used because they were still able to capture the coupling of the electrochemistry and transport phenomena within the electrodes. In the wake of faster

computers, in the past decade, researchers have conducted multi-dimensional simulations. Multi-dimensional models enable better understanding of geometric effects – in particular thermal effects. Multi-dimensional models include the works of Wang and co-workers [Srinivasan, 2003; Smith, 2007; Al Hajjaj, 1999; Nelson, 2002a & 2002b; Karden, 2002; Fang, 2009; Hu, 2011a & 2011b].

Among all the relevant researches, the vast majority of modeling work focuses on validating and demonstrating 1D models for constant load under charging or discharging conditions with relatively low charge / discharge rate (up to 4C). For a battery with 6Ah nominal rating, 1C rate of discharge implies drawing current out of the battery at a rate of 6 Amperes in exactly one hour. For these low charging / discharging rate (lower than 4C), diffusion of Li-ions within the active particles can keep up with the reactions occurring at the surface of the active particle. In other words, Li-ion diffusion resistance within active particles does not play a significant role. In that case, the active particles can be treated as lumped masses. Therefore, for the solid phase concentration field, a simplistic sub-grid scale model can be applied. Those simplifications are adequate for batteries used in electronic devices, because in those conditions, the charging / discharging rate is not significant. However, some researchers, in particular Smith and Wang, pointed out that this assumption is inadequate for those applications in hybrid electric vehicles. In real HEVs, batteries will frequently encounter much higher charging / discharging rates rather than 4C [Smith and Wang, 2006].

1.3. Motivation and Rationale

With advancement of the development of hybrid electric vehicles and the increasing use of Lithium-ion batteries chosen in vehicles, the control of the energy storage system is a topic of central importance. In real world driving cycles, quick and large changes of the load are extremely common. When the load on the battery changes suddenly, the current drawn will change accordingly in a very small time period. However, the battery voltage cannot respond instantaneously, and there must be a time delay causing by the combination of a variety of transport and electrochemical process within the battery. To solve this, so-called equivalent circuit models have been used to describe transient operation of batteries. While these models are efficient, they include little or no insight into the fundamental coupled transport and electrochemical processes within batteries. What is needed is a model that is both computationally efficient (target is much faster than real time) and at the same time, provides an accurate description of the relevant physical processes within the battery.

1.4. Objectives

The objective of this work is to develop a one-dimensional model of a Li-ion battery that computes faster than real time. The model will be validated and then demonstrated for realistic hybrid vehicle driving cycles. The simulations will present an insight into the detailed transport phenomena and electrochemistry with both temporal and spatial resolution. The specific objectives are as follows:

- Understand relevant works through literature review.
- Develop a coupled electrochemical-thermal one-dimensional model of a Li-ion battery.
- Validate the numerical model against published experimental data.
- Explore the numerical model to understand the effect of various operating conditions such as loading cycles, cooling conditions, charging / discharging rate etc. on the performance of the battery.

Chapter 2

METHOD OF RESEARCH

2.1. Lithium-ion Battery Operating Principle

A typical Lithium-ion battery is comprised of three regions: a negative electrode, a positive electrode, and a separator in the middle. A one-dimensional schematic representation is shown in Fig. 3. The negative electrode is a porous structure comprised of lithium carbide particles (Li_xC_6) tightly packed together, while in the positive electrode, the porous structure is comprised of lithium metal oxide particles ($Li_yMn_2O_4$). The most commonly used metal oxides are $LiMn_2O_4$, $LiCoO_2$, or $LiFePO_4$.

The two electrodes are separated by a separator (or ionic conductor) that allows Lithium-ions to migrate across, but prevents any electron transport across it. The entire cell is filled with an electrolyte that occupies the space not occupied by the active particles or the binder and filler (not shown in Fig. 3). The electrolyte is comprised of a lithium salt dissolved in an organic solvent.

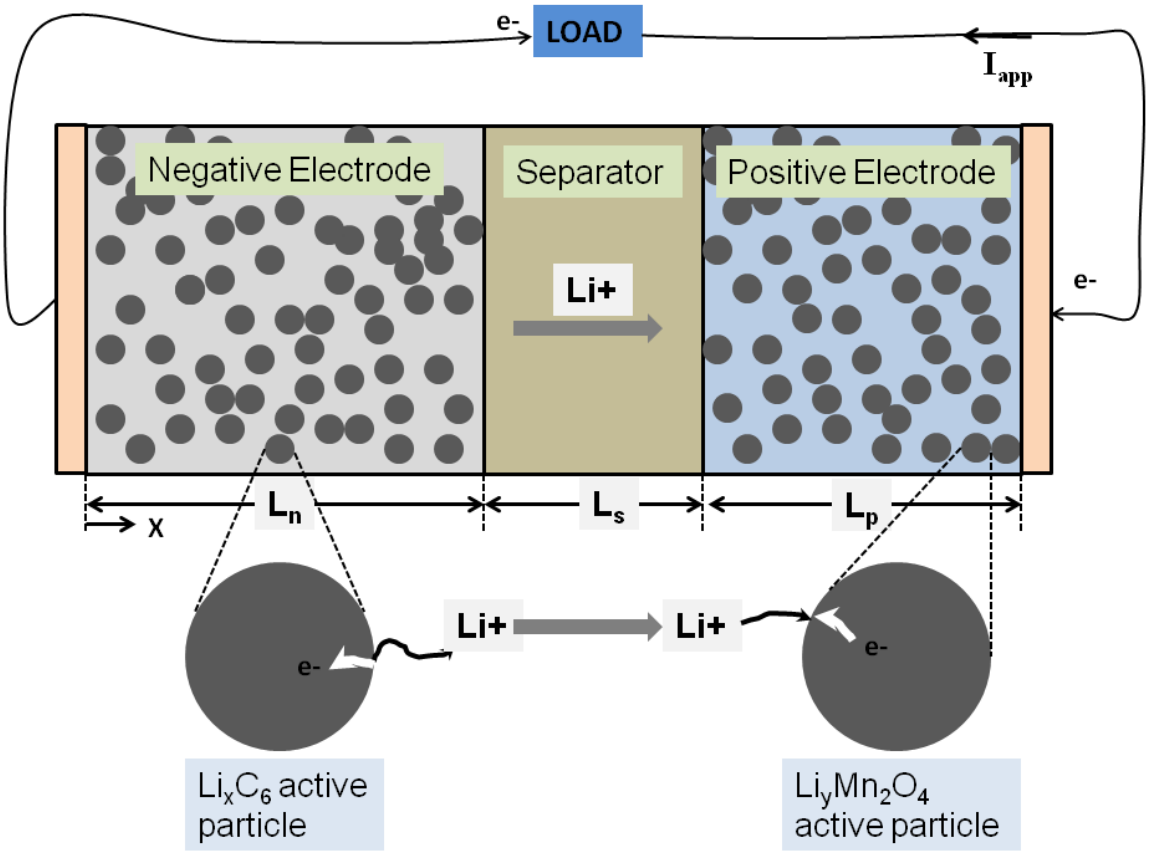

Figure 3 - Schematic one-dimensional representation of a lithium-ion battery under discharge condition

Under normal operating conditions, the following reactions occur at the two electrodes:

Negative Electrode:

$$Li_xC_6 \xrightarrow{discharge} 6C + xLi^+ + xe^- \quad \textbf{(1a)}$$

$$6C + xLi^+ + xe^- \xrightarrow{charge} Li_xC_6 \quad \textbf{(2b)}$$

Positive Electrode:

$$Li_{1-x}Mn_2O_4 + xLi^+ + xe^- \xrightarrow{discharge} LiMn_2O_4 \quad \textbf{(3a)}$$

$$LiMn_2O_4 \xrightarrow{charge} Li_{1-x}Mn_2O_4 + xLi^+ + xe^- \quad \textbf{(4b)}$$

During the discharge process, lithium-ions that are generated at the negative electrode migrate across the separator to the positive electrode, where they recombine with the electrons that flow through the external circuit to form the metal oxide (Fig. 3). The reverse occurs during the charging process.

2.2. Macroscopic Model and Governing Equations

The macroscopic mathematical model employed in this work is the model proposed by Doyle, et al., [2003] and subsequently used by numerous other researchers. While material properties, kinetic constants and other associated properties change depending on the exact chemical composition of the electrodes (as will be pointed out later), the model proposed by Doyle is general enough to be applicable to any Lithium-ion battery since it is based on basic conservation laws. The governing equations are equations of conservation of mass (of Li^+ ions), charge, and energy. Under the assumption of electro-neutrality, the charge conservation equation reduces to a current conservation equation. In general vector notations, the governing equations are written as followed:

$$\frac{\partial}{\partial t}(\varepsilon_e C_e) = \nabla \cdot \left(\varepsilon_e D_e^{eff} \nabla C_e\right) + \frac{1 - t_+}{F} j_{Li} \quad \textbf{(5)}$$

$$\nabla \cdot (\kappa^{eff} \nabla \phi_e) + \nabla \cdot \left(\kappa_D^{eff} \nabla ln C_e\right) = -j_{Li} \quad \textbf{(6)}$$

$$\nabla \cdot (\sigma^{eff} \nabla \phi_e) = j_{Li} \quad \textbf{(7)}$$

where C_e is the molar concentration of Li^+ ions in the electrolyte, ε_e is the volume fraction of the electrolyte, D_e^{eff} is the effective diffusion coefficient of Li^+ ions within the porous electrodes and separator, σ^{eff} is the actual electrical conductivity of particles in electrodes, and j_{Li} is the transfer current.

In order to relate effective diffusion coefficient, D_e^{eff}, to the free-stream diffusion coefficient of Li^+ ions, the Bruggeman relationship is applied:

$$D_e^{eff} = D_e \varepsilon_e^{1.5} \quad \textbf{(6)}$$

where D_e is the free-stream diffusion coefficient of Li^+ ions in the electrolyte, t_+ is the transference number, which represents the charge carried by the Li^+ ions relative to the solvent due to drift, F is the Faraday constant (96487 C/mol), ϕ_e, ϕ_s are the electrolyte (ionic) and solid (electronic) phase potentials, and κ^{eff} and κ_D^{eff} are the effective ionic and diffusional conductivities, respectively.

The diffusional conductivity (electrical conductivity in the electrolyte phase due to diffusion of lithium-ions) is related to the ionic conductivity (electrical conductivity in the electrolyte phase due to drift or electro-migration of lithium-ions) by the following relationship:

$$\kappa_D^{eff} = \frac{2RT\kappa^{eff}}{F}(t_+ - 1)(1 + \frac{\partial ln f}{\partial ln C_e}) \quad \textbf{(7)}$$

where R is the universal gas constant (8.314 J/mol/K), T is the absolute temperature, and f is the mean molar activity coefficient (assumed to be constant in this study due

to lack of better information).

Due to electrochemical reactions at the surface of the active particles, electrons are transferred to the solid (or electronic) phase, while positively charged lithium-ions are transferred to the electrolyte phase. Since current flow is equivalent to electron or Li^+ flow, the electrochemical reaction results in a so-called transfer current, which is essentially the charge equivalent of the numbers of moles generated by the electrochemical reaction.

The transfer current depends on a number of factors, such as the lithium-ion concentrations both in the solid and electrolyte phase, the temperature, and the surface overpotential. It is customary to express the rate of generation of this current using the Butler-Volmer kinetic:

$$j_{Li} = a_s i_0 \left[exp\left\{ \frac{\alpha_a F}{RT} \left(\eta - \frac{R_{SEI}}{a_s} j_{Li} \right) \right\} - exp\left\{ -\frac{\alpha_c F}{RT} \left(\eta - \frac{R_{SEI}}{a_s} j_{Li} \right) \right\} \right] \quad \textbf{(8)}$$

where a_s is the active surface area to volume ratio of the electrode, and i_0 is the exchange current density, which can be written as:

$$i_0 = k_0 C_e^{\alpha_a} \left(C_{s,max} - C_{s,surface} \right)^{\alpha_a} C_{s,surface}^{\alpha_c} \quad \textbf{(98)}$$

where $C_{s,max}$ is the maximum concentration of lithium-ions in the solid phase, $C_{s,surface}$ is the concentration of lithium-ions at the surface of the active particle (i.e., at the solid-electrolyte interface), and $C_{s,max}$ is the maximum concentration of lithium-ions in the solid phase, α_a and α_c are the kinetic constants that depend on the chemical composition of the electrodes and the electrochemical reactions that occur on the active particles, and R_{SEI} is the resistance due to irreversible film

formation at the solid-electrolyte interface.

It is generally believed that these films form in all batteries during the initial assembly process, but grow only if the battery is subjected either to extremely high rates of charge / discharge and / or deep discharge. Aging of the battery is often attributed to growth of this film [Christensen and Newman, 2003] at the anode, which causes increase in the internal resistance of the battery. In the present model, although there is scope to include the effect of this film, as indicated by Eq. 8, due to lack of understanding of the exact mechanism of film formation and growth, this resistance is neglected in this preliminary work.

η, detonating the overpotential, drives the electrochemical reactions, and can be written as:

$$\eta = \phi_s - \phi_e - U \tag{10}$$

where U is the open circuit potential (or equilibrium potential). The open circuit potential is dependent on the chemical composition of the electrode, its state of charge / discharge, and temperature.

The conservation of energy is given by

$$\frac{\partial}{\partial t}\left(\rho c_p T\right) = \nabla \cdot \left(k^{eff} \nabla T\right) + \dot{q} \tag{91}$$

where conduction is assumed to be the only mode of heat transfer. ρ is the average density over all phases, c_p is the average specific heat capacity over all phases, k^{eff} is the average thermal conductivity over all phases, and $\dot{q}$ is the volumetric heat generation rate within the battery.

The volumetric heat generation rate is a net result of three effects: Joule heating,

irreversible heat generation and reversible heat generation. It can be expressed by the following:

$$\dot{q} = a_s j_{Li}\eta + a_s j_{Li} T\left(\frac{\partial U}{\partial T}\right) + \sigma^{eff}\nabla\phi_s \cdot \nabla\phi_s + \kappa^{eff}\nabla\phi_e \cdot \nabla\phi_e + \kappa_D^{eff}\nabla\phi_e \cdot \nabla ln C_e \quad \textbf{(102)}$$

The first term on the right hand side of Eq. 12 represents irreversible heating, and the second term represents reversible heating. The third term represents Joule heating due to the transport of electrons within the two electrodes, while the last two terms represent Joule heating due to the transport of ions within the entire battery.

2.3. Boundary Conditions

The boundary conditions for the governing conservation equations (Eq. 3 – 5 & 11) are well known [Doyle, 1993; Smith, 2006; Srinivasan, 2003], and are summarized in Table 1.

2.4. Sub-Grid Scale Active Particle Model

Equations 3 - 12, along with the boundary conditions shown in Table 1, represent the macroscopic model that can predict the performance (voltage-current) characteristics of a typical lithium-ion battery for both charge and discharge cycles. However, the model is not complete in terms of closure. This is because the solid-phase concentration at the surface of the active particles, $C_{s,surf}$, (as appearing in Eq. 9) is an unknown, and requires additional equations to determine. While the dependent variables in the macroscopic model, namely C_e, C_s, and T are defined within the entire battery, the solid phase concentration is defined only within the

Table 1 - Summary of boundary conditions required for the solution of governing conservation equations

Dependent Variable	**Negative Current Collector**	**Negative / Separator Interface**	**Positive / Separator Interface**	**Positive Current Collector**
C_e [Eq. 3]	Zero flux	Not needed	Not needed	Zero flux
ϕ_e [Eq. 4]	Zero flux	Not needed	Not needed	Zero flux
ϕ_s [Eq. 5]	Applied current	Zero flux	Zero flux	Applied current
T [Eq. 11]	Isothermal, adiabatic, or Newton cooling	Not needed	Not needed	Isothermal, adiabatic, or Newton cooling

active particles. Since the active particles are much smaller than the characteristic dimensions of the electrodes (active particles have a diameter of less than 1 μm, while the electrodes are several tens of μm thick), they cannot be resolved by the same computational mesh that is employed to solve the macroscopic model Eq. 3 - 12. The active particle is essentially at a length scale much smaller than the grid scale of the macroscopic model. Therefore, any equation (or model) used to describe the lithium-ion concentration field within the active particles is at the so-called sub-grid scale, and is henceforth referred to as the sub-grid scale model.

In reality, the active particles are bound together within the electrodes using filler and binder materials. As a result, even though they start as spherical nanoparticles, once bound together, they form a complex network better represented by sets of overlapping spheres of various diameters. Doyle [1993] proposed treating the active particles as non-interacting perfect spheres of constant radius. In this work, we adopt the same assumption, although, with significantly larger computational effort, it is possible to treat more complex shapes, as demonstrated by Kamarajugadda and Mazumder for fuel cell electrodes [2012], and by Wang and Sastry [2007] for battery electrodes. Under the isolated sphere assumption, and also assuming that there is no variation of any quantity along the surface of the sphere (polar and azimuthal uniformity), the concentration of lithium-ions within a single active particle can be described by an unsteady diffusion equation written in spherical coordinates with only radial dependence:

$$\frac{\partial C_s}{\partial t} = \frac{D_s}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial C_s}{\partial r}\right) \tag{113}$$

where D_s is the diffusion coefficient of lithium-ions within the active particles (or solid phase). Equation 13 requires boundary conditions, which are written as the following equations:

$$\left.\frac{\partial C_s}{\partial r}\right|_{r=0} = 0 \quad \textbf{(124)}$$

which equation represents symmetry at the center of the active particle. At the surface of the active particles ($r = r_s$) a diffusion-reaction flux balance yields

$$-D_s \left.\frac{\partial C_s}{\partial r}\right|_{r=0} = \frac{j_{Li}}{a_s F} \quad \textbf{(13)}$$

The active surface area to volume ratio is dependent on the radius of the active particles and their overall volume fraction. This can be represented in the following equation:

$$a_s = \frac{3\varepsilon_s}{r_s} \quad \textbf{(14)}$$

where ε_s is the volume fraction of the active particles in the electrode.

Since the transfer current, j_{Li}, is a non-linear function of the solid phase concentration C_s, as indicated by Eq. 8 & 9, and Eq. 13 represents a partial differential equation whose boundary condition is non-linear, which can be seen in Eq. 14 & 15. Thus, closed form analytical solution of Equation 13 is not possible. Under the assumption that the mass transport by diffusion within the active particle is much faster than the reaction occurring at its surface (i.e., small mass transport Biot Number), the active particle can be treated as a lumped mass, and the solid phase concentration of lithium-ions at the surface of the active particle is given by the following equation:

$$C_{s,surf} - C_{s,avg} = \frac{-j_{Li}}{5a_s F D_S}\left[1 - exp\left(-\frac{20}{3}\frac{\sqrt{D_s t}}{r_s}\right)\right] \quad \textbf{(15)}$$

where $C_{s,avg}$ is the average solid-phase concentration of lithium-ions within the active particle.

However, the lumped mass approximation breaks down in situations where the load is large, as demonstrated by Smith and Wang [2006]. Therefore, for hybrid vehicle applications, it is desirable to solve Eq. 13 in its original form, which is done in the present work.

Solution of Eq. 13 in its original form can be computationally expensive. This is because of two reasons. First, as discussed earlier, its boundary condition is non-linear. Secondly, the equation has to be solved at each computational node of the macroscopic model. Therefore, if 100 nodes are used in the macroscopic battery model, Eq. 13 has to be solved 100 times because for each cell the transfer current, which goes in as an input to the model via the surface boundary condition shown in Eq. 15, may be different. Smith and Wang [2006] solve this equation using a finite-element method. In this work, Eq. 13 is solved using the finite-difference method with nominally 11 grid points.

2.5. Numerical Algorithm

In this work, the Finite Volume Method (FVM) was used to discretize the governing partial differential equations. In this method, the governing conservation equations are integrated over a finite number of control volumes, called cells. Boundary conditions are implemented using appropriate flux expressions developed

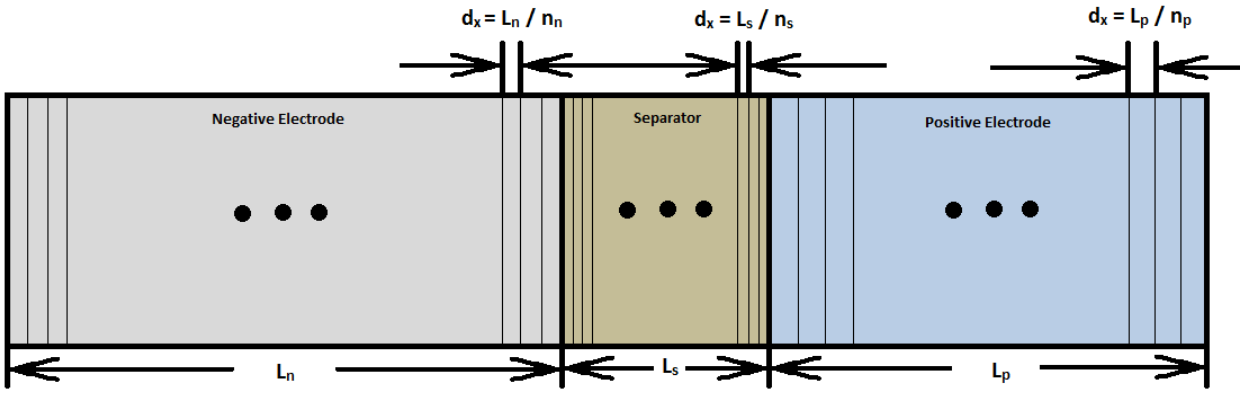

Figure 4 – Simple mesh generation structure

using the boundary condition and Taylor Series expansion. In this work, a non-uniform mesh, as shown in Fig. 4, was used.

For this work, the Finite Volume Method is chosen because it guarantees both local and global conservation irrespective of mesh size, while other methods are inherently non-conservative. Furthermore, it is amenable to handling discontinuities in material properties because the method is basically an integral method.

The finite-volume discretization of the governing equations resulted in a set of non-linear discrete algebraic equations. The non-linearity arises due to the non-linear electrochemical source (as given by Eq. 8). The non-linear source term was firstly linearized using a Taylor Series expansion. The final outcome is a set of linear algebraic equations that assume a tri-diagonal matrix form.

For the discretization of the time derivative, the backward Euler method was used because it is unconditionally stable. Since all of the governing partial differential equations have non-linear source terms, appropriate source term linearization techniques were employed to improve diagonal dominance of the resulting discrete equations sets. For each governing partial differential equations, when discretized and linearized, resulted in a set of tridiagonal equations that were solved using the Thomas algorithm.

In this work, all the partial differential equations were solved sequentially. Coupling between the equations was addressed by using an outer iteration loop over the five governing partial differential equations. This iteration loop was also instrumental in addressing non-linearity in the system.

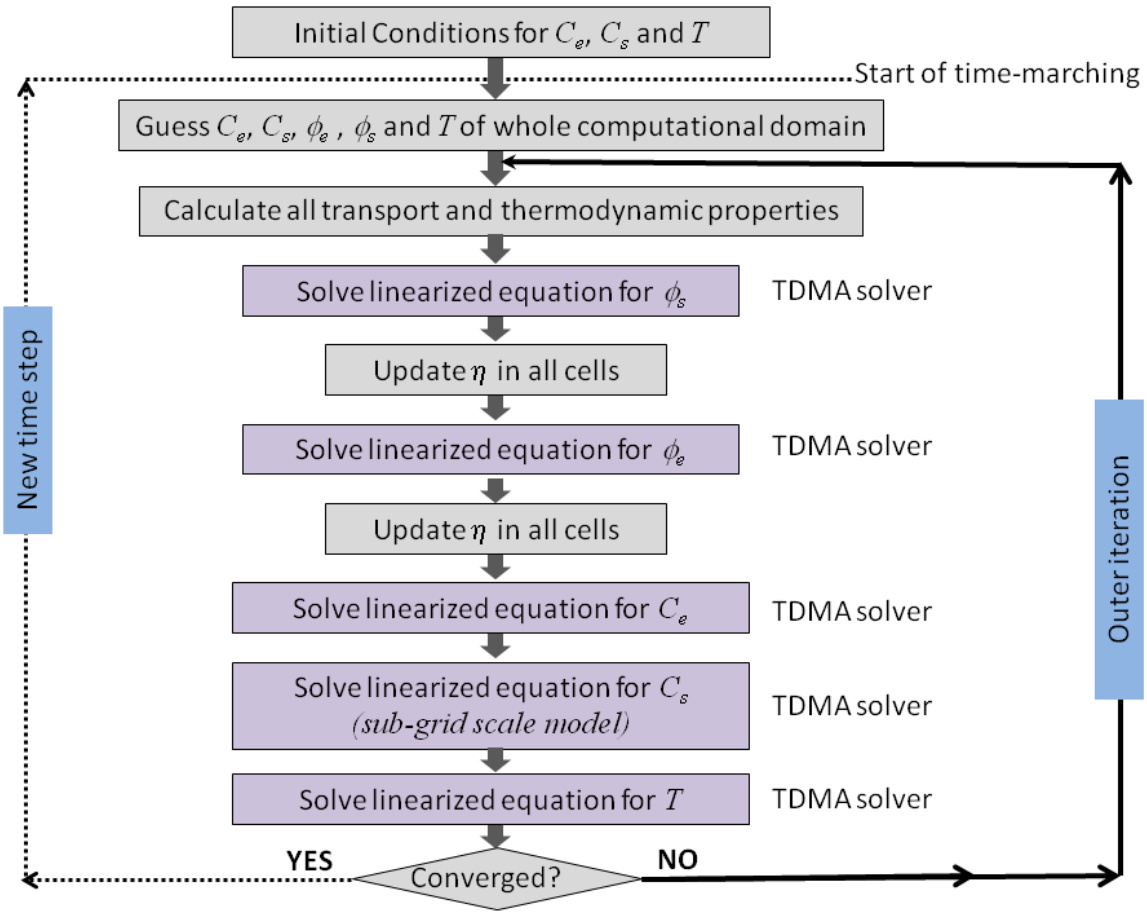

Figure 5 – Solution algorithm employed to solve the governing equation (Algorithm Flow Chart)

Lastly, within each time step, convergence was deemed to have been achieved when the residuals for each conservation equation decreased by at least 4 orders of magnitude. The residual of each partial differential equation was defined as the l2norm (inner product) of the discretized equations without source term linearization. The overall algorithm is depicted in Fig. 5. At each time step, once the solution reaches convergence, the results are post-processed to extract quantities of engineering interest. One of these is the cell voltage, which is given by

$$V_{cell} = \phi_s \mid_{x=L_n+L_s+L_p} - \phi_s \mid_{x=0} - \frac{R_{contact}}{A_{collector}} I_{app} \quad \textbf{(16)}$$

where $R_{contact}$ is the contact resistance between the electrodes and the current collectors, $A_{collector}$ is the collector plate surface area, and I_{app} is the applied current (drawn current) or the load.

The afore-mentioned numerical procedure was implemented using MATLAB, and the MATLAB code developed for this work is included in Appendix.

Chapter 3

RESULTS AND DISCUSSION

3.1. Model Validation

A validation study was first undertaken. In this study, the entire charge and discharge cycle of a $Li_xC_6 — Li_yMn_2O_4$ battery was simulated. The battery considered here has a nominal rating of 6 Ah, and a charge / discharge rate of 1C was used. Experimental data is available for the same set of conditions. Other necessary geometric parameters, material properties, and operating conditions are summarized as followed in Table 2. The data presented in Table 2 are extracted from Smith and Wang [2006], who also provide the experimental data used for the validation study. The only data that were not directly extracted out of Smith and Wang [2006] are the two kinetic constants. The values reported in Table 2 for these constants were estimated to obtain an exchange current density of 36 A/m^2 in the negative electrode and 26 A/m^2 in the positive electrode, as reported by Smith and Wang [2006].

In addition to the data summarized in Table 2, Smith and Wang [2006] also provided curve-fits to experimental measurements for the ionic conductivity and the

open circuit potential. The free stream ionic conductivity (electrical conductivity of the electrolyte phase due to drift of ions) in S/m is given by Smith and Wang [2006]:

$$\kappa = 15.8e-4C_e\, exp\left[0.85\left(\frac{C_e}{1000}\right)^{1.4}\right] \quad \textbf{(17)}$$

where C_e is the molar concentration of Li^+ ions in the electrolyte.

The open circuit potential of the two electrodes was also fitted to experimental data [Smith, 2006] to yield:

$$\begin{aligned} U_n = {} & 8.00229 + 5.0647x - 12.578x^{0.5} - (8.6322e-4)x^{-1} \\ & + (2.1765e-5)x^{\frac{3}{2}} - 0.46016\, exp[15(0.06-x)] \\ & - 0.55364\, exp[-2.4326(x-0.92)] \end{aligned} \quad \textbf{(180)}$$

$$\begin{aligned} U_p = {} & 86.681y^6 - 357.7y^5 + 613.89y^4 - 555.65y^3 + 281.06y^2 \\ & - 76.648y - 0.30987\, exp(5.657y^{115}) + 13.1983 \end{aligned} \quad \textbf{(19)}$$

where x, y are the stoichiometry in the negative and positive electrodes, respectively.

The open circuit potentials computed using Equation 20 & 21 are in Volts. For the external contact resistance, $R_{contact}$, a value of 20 $\Omega\ cm^3$ was used. Although the present model has the ability to predict the temperature field, for the validation purpose, a constant temperature of 288K was used.

In order to compute the voltage characteristics of the entire charge / discharge cycle, the calculations were performed for a net duration of 3800 seconds, with a time step of 1 second. Within each time step, it took approximately 68-72 iterations to achieve convergence by 4 orders of magnitude. A typical convergence plot is shown in Fig. 6.

Table 2 - Values of various parameters for Li_xC_6—$Li_yMn_2O_4$ battery simulated in the present work – geometry and mesh parameters

Parameter	Negative Electrode	Separator	Positive Electrode
Geometry and Mesh			
Thickness, L_n, L_s, L_p	50 μm	25.4 μm	36.4 μm
Active Particle Radius, r_s	1 μm	N / A	1 μm
Electrolyte Volume Fraction, ε_e	0.332	0.5	0.33
Active Particle Volume Fraction, ε_s	0.58	N / A	0.5
Number of Control Volumes (Nominal)	50	25	36
Initial Sate of Battery			
Maximum Solid Phase Concentration, $C_{s,max}$	16.1e3 mol/m^3	N / A	23.9e3 mol/m^3
Stoichiometry at 0% State of Charge, x, y	0.126	N / A	0.936
Stoichiometry at 100% State of Charge, x, y	0.676	N / A	0.442
Electrolyte Concentration, $C_e\|_{t=0}$	1.2e3 mol/m^3	1.2e3 mol/m^3	1.2e3 mol/m^3
Kinetic and Transport Properties			
Kinetic Constant, k_0	1.38e-4 (A/m^2)(m^3/mol)$^{3/2}$		0.64e-4 (A/m^2)(m^3/mol)$^{3/2}$
Charge Transfer Coefficients, α_a, α_c	0.5, 0.5		0.5, 0.5
SEI Film Resistance, R_{SEI}	0		0
Lithium-ion Diffusion Coefficient in Solid Phase, D_s	2e-16 m^2/s		3.7e-16 m^2/s
Lithium-ion Diffusion Coefficient in Electrolyte, D_e	2e-10 m^2/s	2e-10 m^2/s	2e-10 m^2/s
Solid Phase Electrical Conductivity, σ	100 S/m		10 S/m
Electrolyte Phase Ionic Conductivity, κ	[Eq. 19]	[Eq. 19]	[Eq. 19]
Transference Number, t_+	0.363	0.363	0.363
Open Circuit Potential, U	[Eq. 20 & 21]		[Eq. 20 & 21]
Density, ρ	2500 kg/m^3	1200 kg/m^3	1500 kg/m^3
Thermal Conductivity, k^{eff}	5 W/m/K	1 W/m/K	5 W/m/K
Specific Heat Capacity, c_p	700 J/kg/K	700 J/kg/K	700 J/kg/K

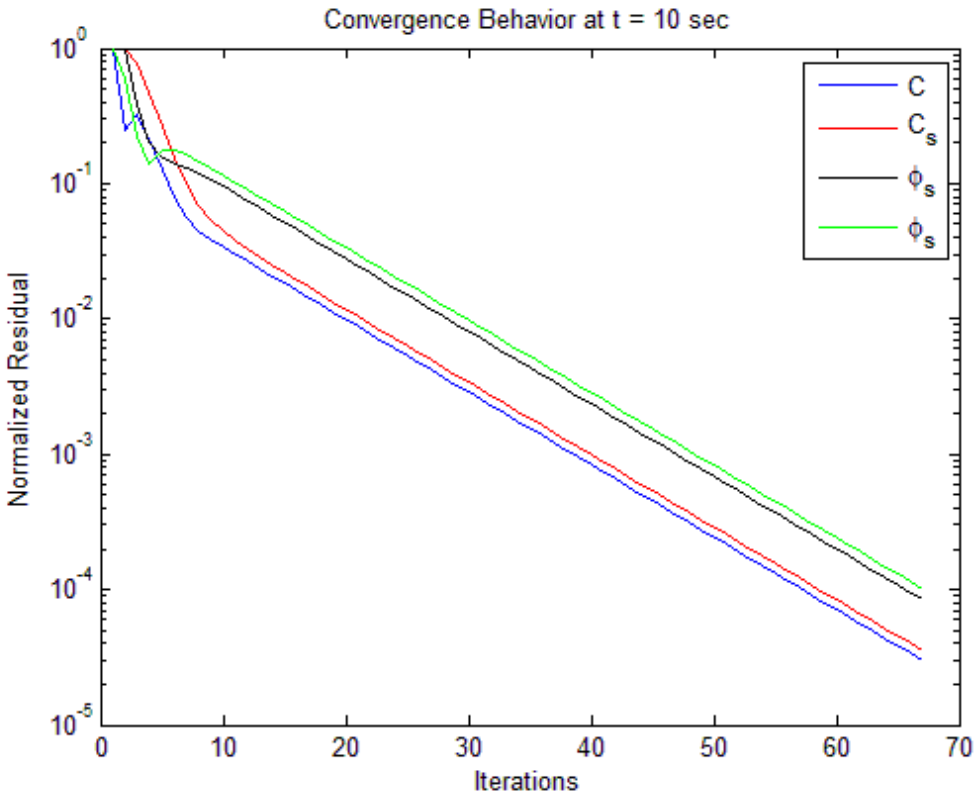

Figure 6 - Convergence behavior at the tenth time-step during a discharging cycle

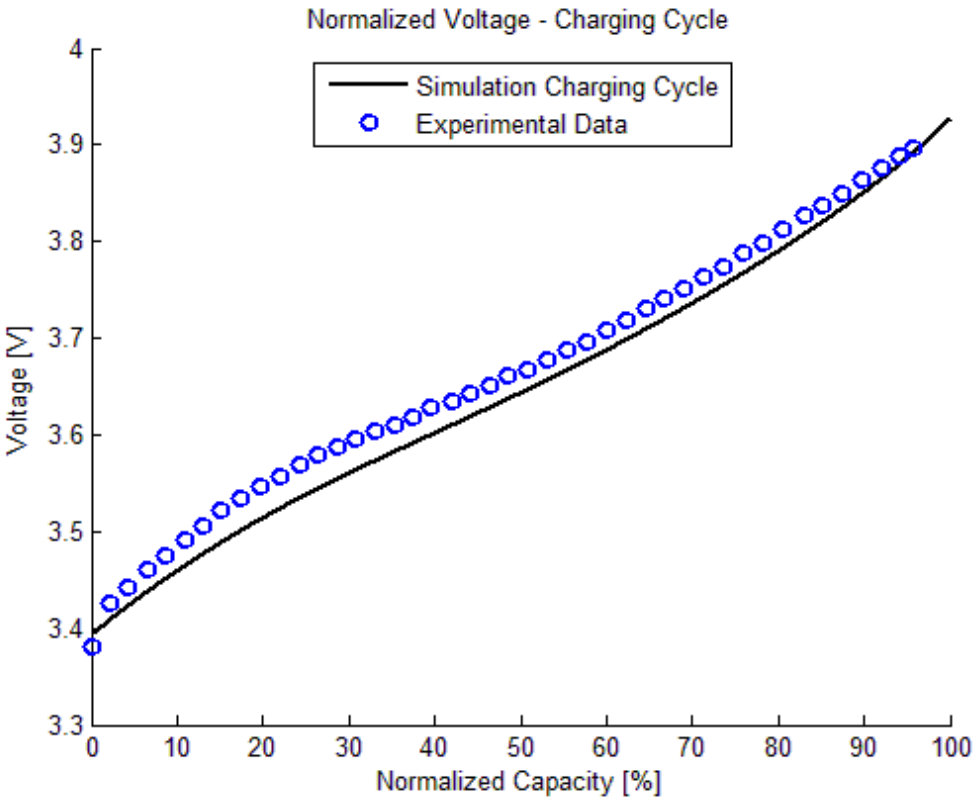

Figure 7 - Computed cell voltage versus measured cell voltage [Smith, 2006] for the entire charge cycle of a Li_xC_6— $Li_yMn_2O_4$ battery with 1C charging rate

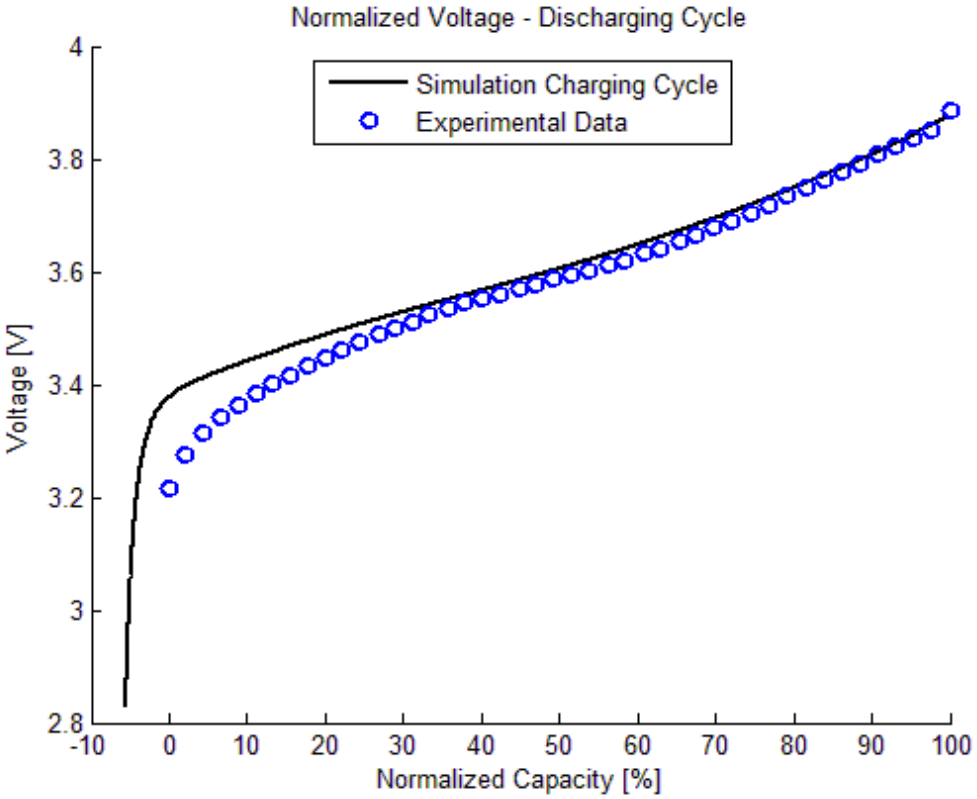

Figure 8 - Computed cell voltage versus measured cell voltage [Smith, 2006] for the entire charge cycle of a Li_xC_6— $Li_yMn_2O_4$ battery with 1C charging rate

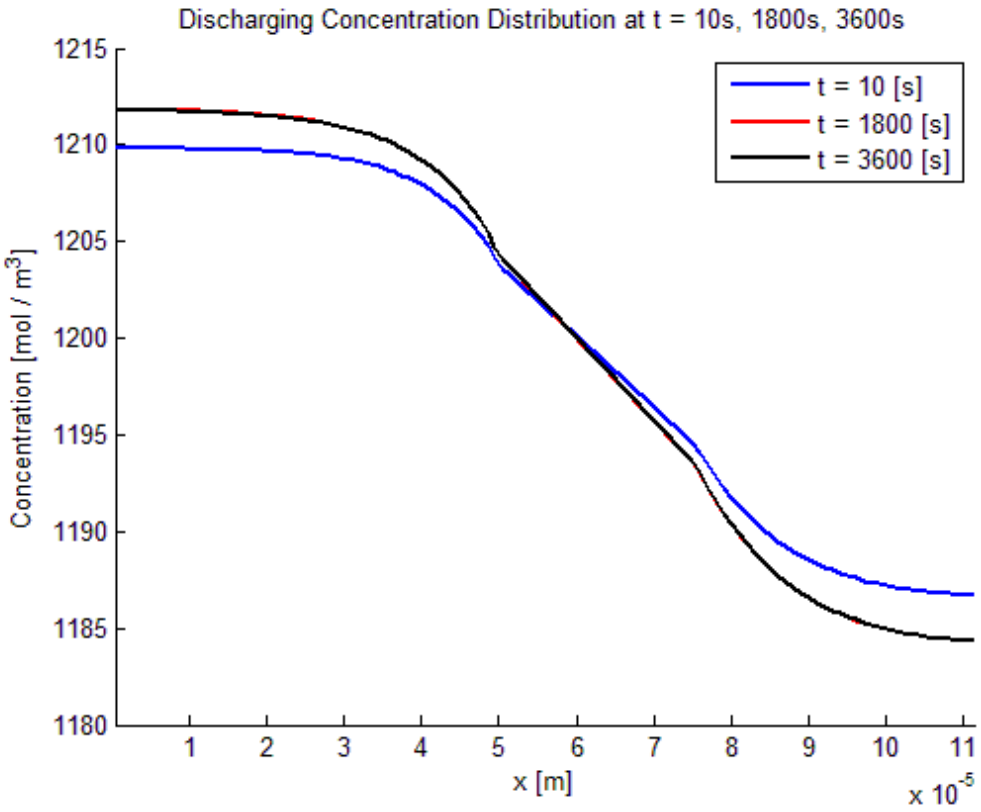

Figure 9 - Spatio-temporal evolution of Electrolyte Phase Concentration of Li+ ion during the beginning, half-way and end of the discharge process with 1C discharging rate

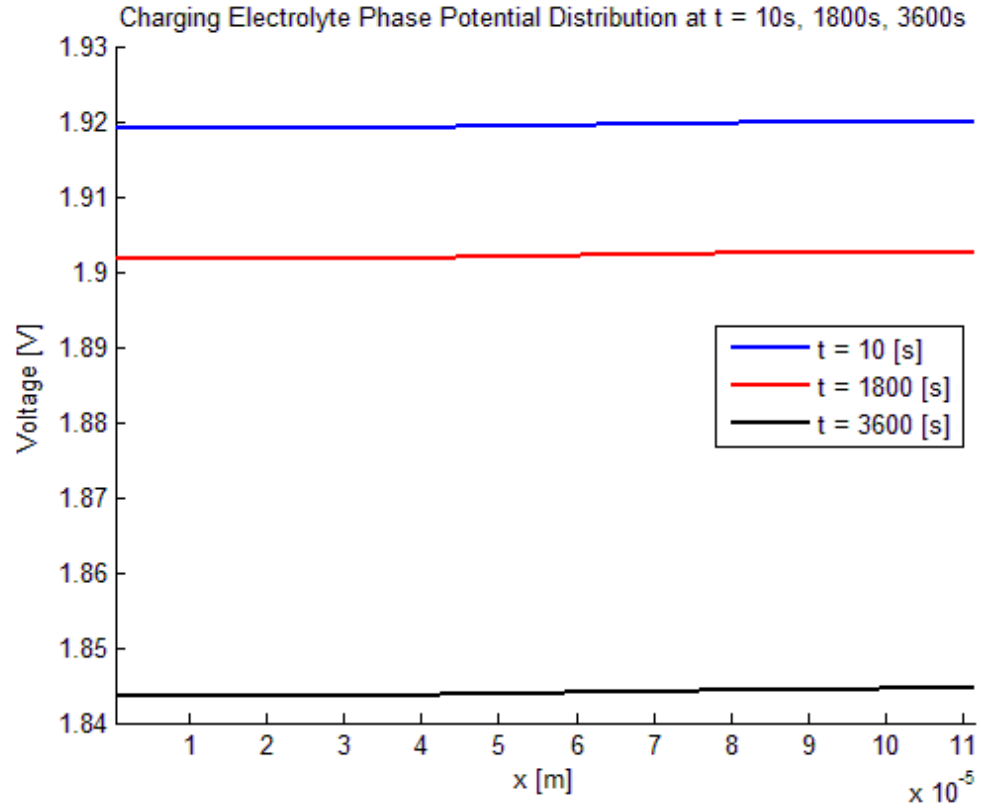

Figure 10 - Spatio-temporal evolution of Electrolyte Phase Potential of Li+ ion during the beginning, half-way and end of the discharge process with 1C discharging rate

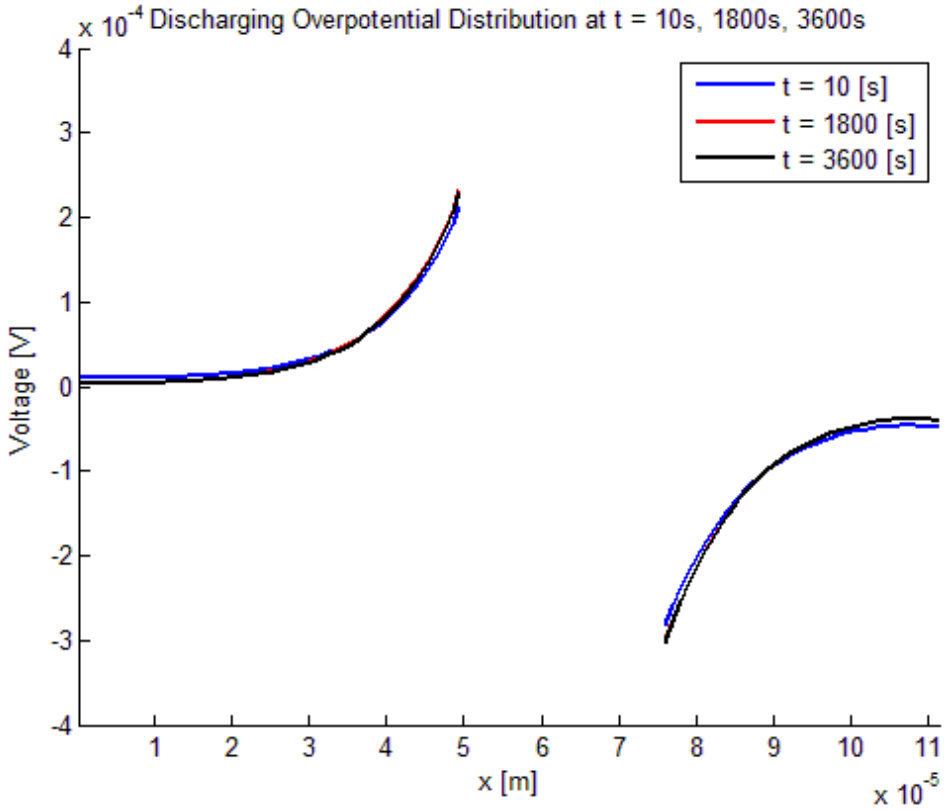

Figure 11 - Spatio-temporal evolution of Overpotential of Li+ ion during the begging, half-way and end of the discharge process with 1C discharging rate

Figures 7 and 8 show the comparison of the computed cell voltage with measured voltage for the entire charge and discharge cycles with a charge / discharge rate of 1C. The computed results agree quite well with the experimental data except toward the extreme end of the discharging cycle (lower than 20%), where the predicted voltage drop occurs slightly later than the measured voltage drop, and at a much sharper rate. This discrepancy could be due to a number of factors including uncertainties in material properties and kinetic constants, neglecting the formation and growth of the SEI film, and thermal effects, and multi-dimensional effects.

Since the present model resolves the battery both temporally and spatially, it is instructive to carefully examine the evolution of some of the dependent variables. Figures 9,10 and 11 show spatio-temporal evolution of lithium-ion concentration in the electrolyte phase, the electrolyte phase electric potential, and the overpotential, respectively.

Figure 9 shows that the lithium-ion concentration distribution changes by a small amount between 1800 seconds and 3600 seconds. There is notable change of Li-ion concentration difference between 10 seconds and 1800 second. Further examination of the change of Li-ion concentration over the discharging cycle revealed that the concentration stops changing around 60 seconds, and after that, it decreases only slightly over the discharging cycle. It is believed that the ability of the negative electrode of the battery in releasing Li-ions does not change a lot. As the concentration difference of Li^+ builds up, the rate of diffusion Li^+ increases. Finally, equilibrium between the rate of release of Li^+ and the diffusion rate of Li^+

is reached, manifesting in a steady-state for Li^+. After the steady state, the ability of battery to release Li^+ decreases slowly, so that the concentration of Li^+ decreases accordingly. In the end, when the battery has no ability to release more Li^+, it is completely "dead."

Figures 10 and 11 present the increase of electrolyte phase potential and the change of overpotential at three time instants during the whole discharging cycle. Overall, the electrolyte phase potential increases uniformly over the time. This is because the ionic conductivity of the entire cell is quite high, resulting in a small potential variation. The overpotential, though quite small at 1C discharge rate, shows significant spatial variation. At the negative electrode, towards the end of the discharge cycle, it rises rapidly. This is consistent with the sharp drop in the electrolyte phase concentration near the negative-separator interface.

The simulation of an entire charge cycle or an entire discharge cycle (3800 time steps with a time step of 1s) required about 2240.2 seconds on a laptop with a 2.3 GHz Intel i7 processor. Therefore, the whole calculation process took almost 3/5 of the real time. The speed is not as fast as expected to be 1/10 of real time, but it is still meeting the target of building a faster-than-real time simulation. More optimization can be applied to improve the calculation speed.

3.2. Demonstration for Hybrid Electric Vehicle Driving Cycle

In order to demonstrate the model for real-time control applications, a driving cycle, typical of that of a hybrid electric vehicle (HEV) was simulated. Current (load) pulses as large as 10C and lasting over only 10 seconds was used. The load cycle was

commenced with the battery at 100% state of charge. Figure 12 shows the computed voltage response of the battery subjected to the prescribed pulse load. The predicted voltage shows the expected behavior, and is able to cope up with the rapidly changing load. The 900-second drive cycle simulation required about 521.3 seconds of CPU time on a laptop with a 2.3 GHz Intel i7 processor. Among those, about 10 seconds was used for writing files and plotting real-time graph as post-processing. This implies that the execution time of the current model is still under 3/5 of the real time.

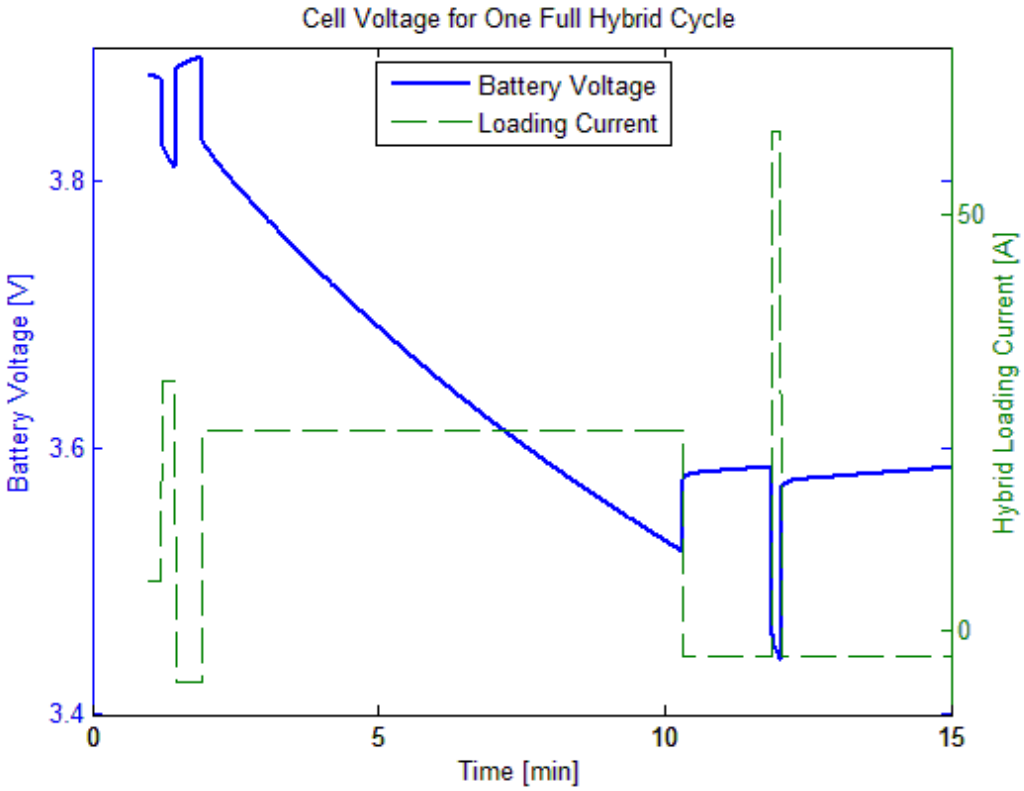

Figure 12 - Computed voltage response of a 6Ah battery subjected to an arbitrary load (current) over a 15-minute period. Positive current denotes discharge, while negative current denotes charging process.

3.3. Thermal Effects

In order to demonstrate the model's ability to simulate thermal effects, the same load cycle considered in the preceding study was now simulated but with the inclusion of the energy equation. Prior to performing the simulation, however, the thermal properties of the materials had to be estimated. Table 2 already shows the thermal properties used for the simulations. These were extracted from Srinivasan and Wang [2003]. These authors also provided curve-fits for the rate of change of the open circuit potential with temperature (as appearing in Eq. 11) for both Li_xC_6 and $Li_yMn_2O_4$ electrodes. During a discharge cycle, reaction (Eq. 1) is endothermic, while reaction (Eq. 2) is exothermic. The net effect of the two reactions, however, is to generate heat irrespective of charge or discharge. This is clearly evident if the battery is operated under adiabatic conditions. Figure 13 shows the temperature evolution of the battery under such conditions. A 4.5K rise in temperature is observed during the 15-minute hybrid drive cycle considered earlier. Although not shown here, even with Newton cooling at one of the ends, the temperature distribution was found to be very uniform because the battery is only 111 μm thick and the resulting conduction resistance across it is very small. To truly understand thermal effects, multi-dimensional modeling is warranted since the temperature variations are dominant along the surface of the battery rather than across it.

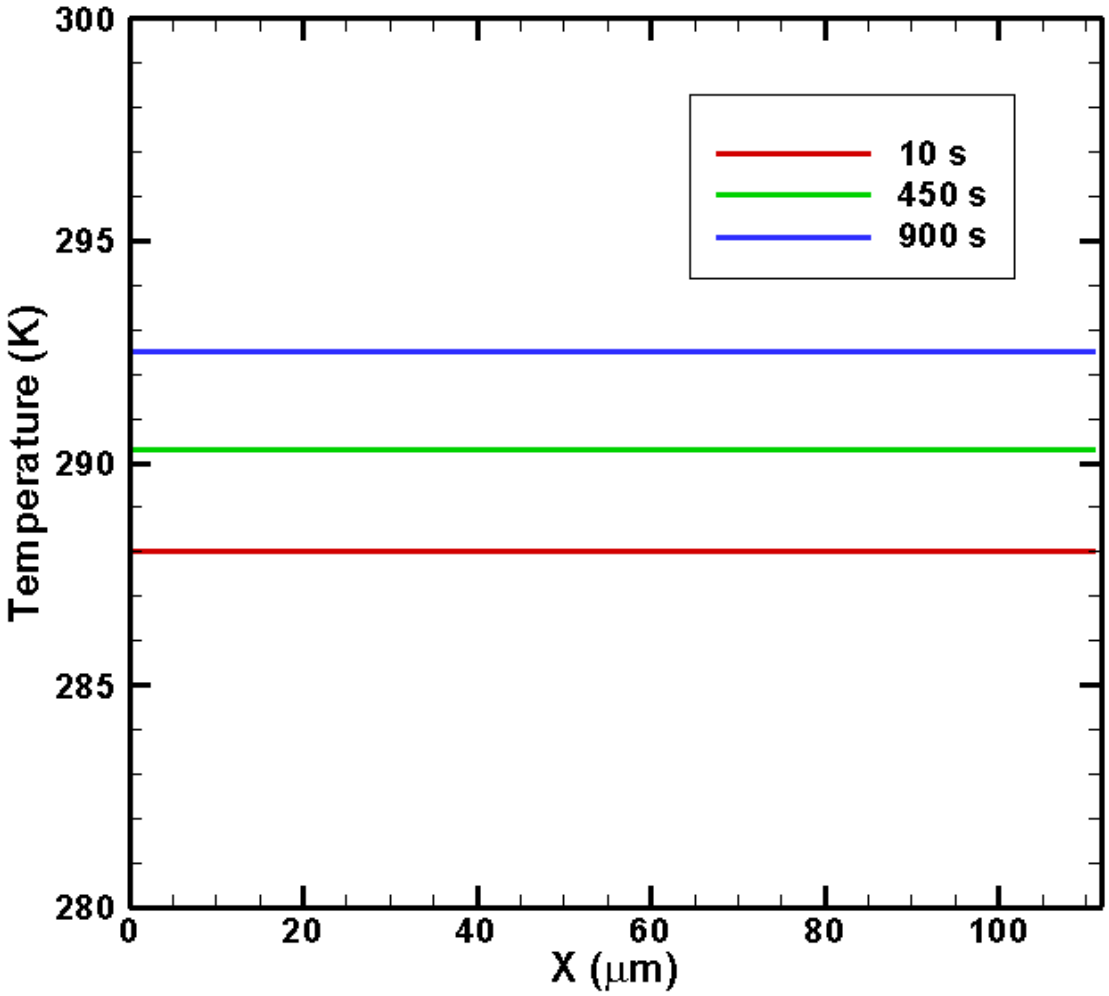

Figure 13 - Spatio-temporal evolution of temperature within the battery under adiabatic conditions during a 15 minute hybrid drive cycle

Chapter 4

CONCLUSIONS AND FUTURE WORK

4.1. Summary and Conclusions

A one-dimensional faster than real time computational model, for predicting the electrochemical-thermal behavior of a lithium-ion battery subjected to a prescribed load, was successfully developed. The model completely resolves both time and space across the battery and is, therefore, capable of capturing any spatial non-uniformity of relevant quantities such as concentrations, electric potentials, and temperature within the battery. Although the model is validated and demonstrated here for a Li_xC_6—$Li_yMn_2O_4$ battery, in principle, it is general enough to be applicable to any lithium-ion battery provided with the material properties and kinetics are appropriately characterized and known. The model was validated against experimental measurements for the entire charge and discharge cycle of a Li_xC_6 — $LiMn_2O_4$ battery, and was able to accurately predict the voltage characteristics. It was then successfully demonstrated for a load input typical of that of an HEV automotive drive cycle.

The computational efficiency of the model was noteworthy. A 900-second drive cycle was computed in less than 525 seconds of CPU time on a laptop, indicating that the model can be integrated with real-time control algorithms.

4.2. Future Work

For the simulation demonstrated in this work, there are many limitations. The model does not include sub-models to capture "aging" phenomenon. This so-called "aging" phenomena is still a challenge for researchers, and to accurate simulate it, more work is needed, so that the life cycle of a battery can be predicted.

Moreover, an extension of the model to complex three-dimensional geometries is warranted, especially to fully understand thermal effects. Since temperature rise of the battery is intimately connected to both the electro-chemistry and transport processes, an accurate model of a battery must include appropriate thermal effects.

REFERENCE

Al Hajjaj, S., H. Maleki, J. Hong, and J. Selman. (1999) "Thermal Modeling and Design Considerations of Lithium-ion Batteries." *Journal of Power Sources* 83: 1-8.

Christensen, J., and J. Newman. (2003) "Effect of Anode Film Resistance on the Charge / discharge Capacity of a Lithium-Ion Battery." *Journal of The Electrochemical Society* 150.11: A1416-1420.

Doyle, Marc, T. F. Fuller, and J. Newman. (1993) "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell." *Journal of The Electrochemical Society* 140.6: 1526.

Fang, W., O. J. Kwan, C. Wang, and Y. Ishikawa. (2009) "Modeling of Li-ion Battery Performance in Hybrid Electric Vehicles." *SAE Technical Paper* 01.1388: n. pag.

"HEV Vehicle Battery Types." (2013) *ThermoAnalytics :* Web. <http://www.thermoanalytics.com/support/publications/batterytypesdoc.html>

Hu, Y., and S. Yurkovich. (2011a) "Linear Parameter Varying Battery Model Identification Using Subspace Methods." *Journal of Power Sources* 196: 2913-923.

Hu, Y., S. Yurkovich, Y. Guezennec, and B.J. Yurkovich. (2011b) "Electro-thermal Battery Model Identification for Automotive Applications." *Journal of Power Sources* 196.1: 449-57.

Hughes, Adrian Kingsley. (2012) "Could the IPhone 5 Push Li-Ion Batteries to Their Limit?" Editorial. *ZDNet*. N.p., 20 July 2012. Web. 22 Feb. 2013. <http://www.zdnet.com/could-the-iphone-5-push-li-ion-batteries-to-their-limit-7000001312/>.

Kamarajugadda, S., and S. Mazumder. (2012) "Generalized Flooded Agglomerate Model for the Cathode Catalyst Layer of a Polymer Electrolyte Membrane Fuel Cell." *Journal of Power Sources* 28: 328-39.

Karden, E., S. Buller, and R. DeDoncker. (2002)"A Frequency-domain Approach to Dynamical Modeling of Electrochemical Power Sources." *Electrochimica Acta* 47.13-14: 2347-356.

Miller, Josh. (2011) "Toyota Hybrid to Get Lithium-ion Batteries." Editorial. *The Car Tech Blog*. N.p., 31 Jan. 2011. Web. 22 Feb. 2013. <http://reviews.cnet.com/8301-13746_7-20030072-48.html>.

Nelson, P., D. Dees, K. Amine, and G. Henriksen. (2002a) "Modeling Thermal Management of Lithium-ion PNGV Batteries." *Journal of Power Sources* 110: 349-56.

Nelson, P., I. Bloom, K. Amine, and G. Henriksen. (2002b) "Design Modeling of Lithium-ion Battery Performance." *Journal of Power Sources* 110: 437-44.

Ning, G., R. White, and B. Popov. (2006) "A Generalized Cycle Life Model of Rechargeable Li-ion Batteries." *Electrochimica Acta* 51.10: 2012-022.

Ramadass, P., B. Haran, R. E. White, and B. N. Popov. (2003) "Mathematical Modeling of the Capacity Fade of Li-ion Cells." *Journal of Power Sources* 123.2: 230-40.

Smith, K., C. Rahn, and C. Wang. (2007) "Control Oriented 1D Electrochemical Model of Lithium-ion Battery." *Energy Conversion and Management* 48.9: 2565-578.

Smith, Kandler, and Chao-Yang Wang. (2006) "Solid-state Diffusion Limitations on Pulse Operation of a Lithium-ion Cell for Hybrid Electric Vehicles." *Journal of Power Sources* 161.1: 628-39.

Srinivasan, Venkat, and C. Y. Wang. (2003) "Analysis of Electrochemical and Thermal Behavior of Li-Ion Cells." *Journal of The Electrochemical Society* 150.1: A98.

Thomas, Justin. (2007) "Make Lithium-Ion Batteries For Hybrids." Editorial. *TreeHugger*. N.p., 13 Apr. 2007. Web. 22 Feb. 2013. <http://www.treehugger.com/gadgets/nissan-and-nec-to-make-lithium-ion-batteries-for-hybrids.html>.

Wang, Chia-Wei, and Ann Marie Sastry. (2007) "Mesoscale Modeling of a Li-Ion Polymer Cell." *Journal of The Electrochemical Society* 154.11: A1035.

APPENDIX

Please see the following pages for the MATLAB script files. Thanks!

```
% main.m

% ************************************************************************

% Lithium-ion Battery Simulation
% 1D-model
% Testing Version

% Main Program

% ************************************************************************

clc;clear;

%% Set up grid parameter, material properties, and initial conditions

    % -------------------------------------------------------------------------
    % Charging / Discharging Condition
    charge = true;              % Initially Assume Charging Cycle
    % -------------------------------------------------------------------------
    % Hybrid Cycle
    hybrid_cycle = false;
    % -------------------------------------------------------------------------
    % Simulation Time
    time = 4000;                % Simulation End Time [UNIT: seconds]
    % -------------------------------------------------------------------------

% Load subscript
settings

%% Run iteration loops
% Start of time marching loop

% Timer - Start
TIMER = cputime;

for it = 1 : time_max

    % Start of iteration at each time step --------------------------------
    for iter = 1: iter_max

        % Calculate equilibrium voltage
        loop_eqpot

        % Solve solid (electronic) phase electrical potential
        loop_phiS

        % Solve electrolyte phase electrical potential
        loop_phiE

        % Solve Li-ion electrolyte phase concentration
```

```
        loop_conc

        % Solve Li-ion solid phase surface concentration
        loop_concS

        % Check convergence
        loop_check

        if iter == iter_max-1
            display('NOT CONVERGED')
        end
        if converged == true
            display('CONVERGED');
            break;
        end
    end
    % End of time-step iteration loop -------------------------------------

    %Time advance solution
    time_march

    % Output result
    post

% Exit Condition
if cell_voltage < 0
    fprintf('\n\nNegative cell_voltage!\nSimulation Ended!\n\n');
    break;
end

end
% End of time loop

% Timer - End
timeConsumed = cputime - TIMER
%% End of program/simulation

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings.m

% ************************************************************************

% Load all the settings files.

% ************************************************************************

settings_grid
settings_control

settings_material
settings_atCnF
settings_initial

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_grid.m

% *************************************************************************

% Settings File - Simulation Geometry Parameters
% Define default settings
% Settings here can be changed to improve precision

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% *************************************************************************

% Time Step
dt = 1.0e0;                 % Time Step Length [UNIT: seconds]----------------------Test↙
Config

% Grid Parameters
n_n = 50;                   % Number of Finite Volume Nodes - NEGATIVE-------------Test↙
Config
n_s = 25;                   % Number of Finite Volume Nodes - SEPARATOR------------Test↙
Config
n_p = 36;                   % Number of Finite Volume Nodes - POSITIVE-------------Test↙
Config
n_t = n_n + n_s + n_p;      % Total Number of Cells

l_n = 50e-6;                % Length of NEGATIVE Electrode [UNIT: m]
l_s = 25.4e-6;              % Length of SEPARATOR          [UNIT: m]
l_p = 36.4e-6;              % Length of POSITIVE Electrode [UNIT: m]

% *************************************************************************
% Mesh Node Properties

dx_n = l_n / n_n;           % Cell size in NEGATIVE Electrode
dx_s = l_s / n_s;           % Cell size in SEPARATOR
dx_p = l_p / n_p;           % Cell size in POSITIVE Electrode

% -------------------------------------------------------------------------

% Volume and Grid Spacing of Each Cell
vol = zeros( 1, n_t );      % Preallocation
dx = zeros( 1, n_t );       % Preallocation

for i = 1 : n_n             % NEGATIVE
    vol(i) = dx_n;
    dx(i) = dx_n;
end
for i = n_n + 1 : n_n + n_s % SEPARATOR
    vol(i) = dx_s;
    dx(i) = dx_s;
end
for i = n_n + n_s + 1 : n_t % POSITIVE
```

```
    vol(i) = dx_p;
    dx(i) = dx_p;
end

% ---------------------------------------------------------------------

% Cell Center of Each Cells
xc = zeros( 1, n_t );       % Preallocation

% For i = 1
xc(1) = dx_n / 2;

% For i = 2 : n_t
for i = 2 : n_t
    xc(i) = xc(i-1) + ( dx(i-1) + dx(i) ) / 2;
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_control.m

% ************************************************************************

% Settings File - Simulation Control Parameters
% Define default settings

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% ************************************************************************

% Maxium Tolerance
tol_outer = 1.0e-4;

% relax - Change to Lower Value for Higher Charge Rate
relax = 0.5;                % ----------------------------------------------------↙
Test Config

% ------------------------------------------------------------------------

% Loop Control Parameters

TINY = 1e-20;                   % Defination of "tiny"

time_max = time / dt;           % Maxium Time Steps-----------------------------------↙
Test Config
iter_max = 500;                 % Max Number of Iteration Loops-----------------------↙
Test Config
it_inner_max = 1;               % Max Number of Inner Iteration Loops-----------------↙
Test Config
res_max( 1 : 4 ) = TINY;        % Maxium Allowed res

% ------------------------------------------------------------------------

% Property Limitations - Minimum / Maximum Values - Should Not Be Changed

conc_min = 1e-40;           % Minimum Concentration Value [UNIT: mol/m^3]
conc_max = 1e5;             % Maximum Concentration Value [UNIT: mol/m^3]

phiE_min = -10;             % Minimum Electrolyte Phase Potential [UNIT: V]
phiE_max = 10;              % Maximum Electrolyte Phase Potential [UNIT: V]

phiS_min = -10;             % Minimum Solid Phase Potential [UNIT: V]
phiS_max = 10;              % Maximum Solid Phase Potential [UNIT: V]

eta_min = -5e-1;            % Minimum Allowed eta Value
eta_max = 5e-1;             % Maximum Allowed eta Value

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_material.m

% ************************************************************************

% Settings File - Simulation Material Properties
% Define default settings
% All physical properties - won't change until changing materials

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% Units are all in International System of Units (SI)

% ************************************************************************

% ------------------------------------------------------------------------

% Load All Physical Constants
settings_physical

% ------------------------------------------------------------------------

% Volume Fractions, ie. porosity

porosity_n = 0.332;         % Porosity Coefficient of NEGATIVE Electrode [Dimensionless]
porosity_s = 0.500;         % Porosity Coefficient of NEGATIVE Electrode [Dimensionless]
porosity_p = 0.330;         % Porosity Coefficient of POSITIVE Electrode [Dimensionless]

% ------------------------------------------------------------------------

% Volume Fraction of Active Particles in Electrodes

poroActive_n = 0.58;        % Porosity of Active in NEGATIVE Electrode [Dimensionless]
poroActive_p = 0.50;        % Porosity of Active in POSITIVE Electrode [Dimensionless]

% ------------------------------------------------------------------------

% diffusion Coefficient

diffu = 2.6e-10;             % diffusivity for BOTH Electrodes and SEPARATER [UNIT:↙
m^2/s]

% diffusion Coefficient of Li-ion in Solid Active Particles

diffuSolid_n = 2.0e-16;      % Li+ diffusivity in Solid Phase for NEGATIVE Electrode↙
[UNIT: m^2/s]
diffuSolid_p = 3.7e-16;      % Li+ diffusivity in Solid Phase for POSITIVE Electrode↙
[UNIT: m^2/s]

% Transference Number - ASSUMED CONSTANT FOR SIMPLIFICATION

tPlus = 0.363;           % [Dimensionless]
```

```
% Radius of Active Particles in Electrodes

R_s = 1.0e-6;           % At BOTH Electrodes [UNIT: m]

% Transfer Coefficient of Surface Reaction
alpha_a = 0.5;          % At NEGATIVE Electrode (Cathodic) [Dimensionless]
alpha_c = 0.5;          % At POSITIVE Electrode (anodic) [Dimensionless]

% Maxium Concentration of Li-ion on Surface of Particles
concS_max_n = 16.1e3;           % At NEGATIVE Electrode [UNIT: mol/m^3]
concS_max_p = 23.9e3;           % At POSITIVE Electrode [UNIT: mol/m^3]

% Surface Electrochemical Reaction-Rate Constant
kcon_n = 1.38e-4;       % At NEGATIVE Electrode [UNIT: mol^(-3/2) * m^(-1/2) * s^(-1)]
kcon_p = 0.64e-4;       % At POSITIVE Electrode [UNIT: mol^(-3/2) * m^(-1/2) * s^(-1)]

% Actual Electrical Conductivity of Particles
sigma_n = 100;          % At NEGATIVE Electrode [UNIT: s/m]
sigma_p = 10;           % At POSITIVE Electrode [UNIT: s/m]

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_physical.m

% ************************************************************************

% Settings File - Some Relative Physical Constants
% Define default settings
% All physical constants - Never Change

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% Units are all in International System of Units (SI)

% ************************************************************************

% Faraday Constant
FARADAY = 96487;         % [UNIT: C/mol]

% Ideal Gas Constant
RU = 8.314;              % [UNIT: J/mol/K]

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_atCnF.m

% ************************************************************************

% Settings File - Some Additional Material Properties for Each Nodes
% Define default settings
% All physical properties - won't change until changing materials

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% Units are all in International System of Units (SI)

% ************************************************************************

% ------------------------------------------------------------------------

% Prosity at Each Node
porosityNODE = zeros( 1, n_t );             % Preallocation
poroActiveNODE = zeros( 1, n_t );           % Preallocation

for i = 1 : n_n
    porosityNODE(i) = porosity_n;
    poroActiveNODE(i) = poroActive_n;
end
for i = n_n + 1 : n_n + n_s
    porosityNODE(i) = porosity_s;
    poroActiveNODE(i) = 0;
end
for i = n_n + n_s + 1 : n_t
    porosityNODE(i) = porosity_p;
    poroActiveNODE(i) = poroActive_p;
end

% ------------------------------------------------------------------------

% diffusion Coefficient at Each Node
diffuNODE = zeros( 1, n_t );             % Preallocation

for i = 1 : n_t
    diffuNODE(i) = diffu * ( porosityNODE(i)^1.5 );
end

% ------------------------------------------------------------------------

% EffectiveElectronic conductivity at each node
sigmaEffNODE = zeros( 1, n_t );                         % Preallocation
sigmaNODE = zeros( 1 , n_t );

for i = 1 : n_n
    sigmaEffNODE(i) = sigma_n * ( poroActiveNODE(i)^1.0 );
    sigmaNODE(i) = sigma_n;
```

```
end
for i = n_n + n_s + 1 : n_t
    sigmaEffNODE(i) = sigma_p * ( poroActiveNODE(i)^1.0 );
    sigmaNODE(i) = sigma_n;
end

% ************************************************************************

% Transport porosity and diffusion at Faces
porosityFACE = zeros( 1, n_t + 1 );            % Preallocation
poroActiveFACE = zeros( 1, n_t + 1 );          % Preallocation
diffuFACE = zeros( 1, n_t + 1 );                % Preallocation
sigmaEffFACE = zeros( 1, n_t + 1 );               % Preallocation
sigmaFACE = zeros( 1, n_t + 1 );               % Preallocation

% Left Boundary
porosityFACE(1) = porosityNODE(1);
poroActiveFACE(1) = porosityNODE(1);
diffuFACE(1) = diffuNODE(1);
sigmaEffFACE(1) = sigmaEffNODE(1);
sigmaFACE(1) = sigmaNODE(1);

% NEGATIVE
for i = 2 : n_n
    diffuFACE(i) = ( diffuNODE( i - 1 ) + diffuNODE(i) ) / 2;
    porosityFACE(i) = ( porosityNODE( i - 1 ) + porosityNODE(i) ) / 2;
    sigmaEffFACE(i) = ( sigmaEffNODE( i - 1 ) + sigmaEffNODE( i ) ) / 2;
    sigmaFACE(i) = ( sigmaNODE( i - 1 ) + sigmaNODE( i ) ) / 2;
    poroActiveFACE(i) = ( porosityNODE( i - 1 ) + porosityNODE( i ) ) / 2;
end

% NEGATIVE and SEPARATER Boundary
wt = ( 1 / dx( n_n ) ) / ( 1 / dx( n_n ) + 1 / dx( n_n + 1 ) );
avg_p = wt * porosityNODE( n_n ) + ( 1 - wt ) * porosityNODE( n_n + 1 );
avg_d = wt * diffuNODE( n_n ) + ( 1 - wt ) * diffuNODE( n_n + 1 );

porosityFACE( n_n + 1 ) = porosityNODE( n_n ) * porosityNODE( n_n + 1 ) / avg_p;
poroActiveFACE( n_n + 1 ) = 0;%poroActiveFACE( n_n );
diffuFACE( n_n + 1 ) = diffuNODE( n_n ) * diffuNODE( n_n + 1 ) / avg_d;
sigmaEffFACE( n_n + 1 ) = 0;%sigmaEffFACE( n_n );
sigmaFACE( n_n + 1 ) = 0;%sigmaFACE( n_n );

% SEPARATER
for i = n_n + 2 : n_n + n_s
    diffuFACE(i) = ( diffuNODE( i - 1 ) + diffuNODE( i ) ) / 2;
    porosityFACE(i) = (porosityNODE( i - 1 ) + porosityNODE( i ) ) / 2;
    sigmaEffFACE(i) = 0;
    sigmaFACE(i) = 0;
    poroActiveFACE(i) = 0;
end
```

```
% POSITIVE and SEPARATER Boundary
wt = ( 1 / dx( n_n + n_s ) ) / ( 1 / dx( n_n + n_s ) + 1 / dx( n_n + n_s + 1 ) );
avg_p = wt * porosityNODE( n_n + n_s ) + ( 1 - wt ) * porosityNODE( n_n + n_s + 1 );
avg_d = wt * diffuNODE( n_n + n_s ) + ( 1 - wt ) * diffuNODE( n_n + n_s + 1 );

porosityFACE( n_n + n_s + 1 ) = porosityNODE( n_n + n_s ) * porosityNODE( n_n + n_s + 1 )↙
/ avg_p;
poroActiveFACE( n_n + n_s + 1 ) = 0;
diffuFACE( n_n + n_s + 1 ) = diffuNODE( n_n + n_s ) * diffuNODE( n_n + n_s + 1 ) / avg_d;
sigmaEffFACE( n_n + n_s + 1 ) = 0;
sigmaFACE( n_n + n_s + 1 ) = 0;

% POSITIVE
for i = n_n + n_s + 2 : n_t
    porosityFACE(i) = ( porosityNODE( i - 1 ) + porosityNODE( i ) ) / 2;
    poroActiveFACE(i) = (poroActiveNODE( i - 1 ) + poroActiveNODE( i ) ) / 2;
    diffuFACE(i) = ( diffuNODE( i - 1 ) + diffuNODE( i ) ) / 2;
    sigmaEffFACE(i) = ( sigmaEffNODE( i - 1 ) + sigmaEffNODE( i ) ) / 2;
    sigmaFACE(i) = ( sigmaNODE( i - 1 ) + sigmaNODE( i ) ) / 2;
end

% Right Boundary
porosityFACE( n_t + 1 ) = porosityNODE( n_t );
poroActiveFACE( n_t + 1 ) = poroActiveNODE( n_t );
diffuFACE( n_t + 1 ) = diffuNODE( n_t );
sigmaEffFACE( n_t + 1 ) = sigmaEffNODE( n_t );
sigmaFACE( n_t + 1 ) = sigmaNODE( n_t );


% ------------------------------------------------------------------------

% Interfacial Surface Area-Volume Ratio of Particle
sv = zeros( 1, n_t );                          % Preallocation
for i = 1 : n_t
    sv(i) = 3.0 * poroActiveNODE(i) / R_s;         %[UNIT: 1/m]
end

% Surface Electrochemical Reaction-Rate Constant at each node
kconNODE = zeros( 1, n_t );                        % Preallocation
for i = 1 : n_n
    kconNODE(i) = 1.38e-4;       % At NEGATIVE Electrode [UNIT: mol^(-3/2) * m^(-1/2) *↙
s^(-1)]
end
for i = n_n + n_s + 1 : n_t
    kconNODE(i) = 0.64e-4;       % At POSITIVE Electrode [UNIT: mol^(-3/2) * m^(-1/2) *↙
s^(-1)]
end

% Maxium Concentration of Li-ion on Surface of Particles at each node
concS_maxNODE = zeros( 1, n_t );                       % Preallocation
for i = 1 : n_n
```

```
    concS_maxNODE(i) = 16.1e3;           % At NEGATIVE Electrode [UNIT: mol/m^3]
end
for i = n_n + n_s + 1 : n_t
    concS_maxNODE(i) = 23.9e3;           % At POSITIVE Electrode [UNIT: mol/m^3]
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% settings_initial.m

% ************************************************************************

% Settings File - Initial Conditions / Starting Conditions
% Define default settings
% Testing Configuration Variables

% ALL CAPITAL VARIABLES NEVER CHANGE IN LOOPS

% Units are all in International System of Units (SI)

% ************************************************************************

% ------------------------------------------------------------------------

% Boundary Conditions and Other Parameters
i_app_D = 6;                % 1-C Discharge Rate
i_app_C = -6;               % 1-C Charge Rate

if charge == true           %↙
-------------------------------------------------------------------
    i_app = i_app_C;        %
else                        % Charging / Discharging Condition Switch
    i_app = i_app_D;        %
end                         %↙
-------------------------------------------------------------------

resistance = 20e-4;         % External Contact Resistance [UNIT: Ohm * m^2]
capacity = 6;               % External Circuit Capacity [UNIT: Ah = Amp * h]
i_app_cum = 0;              % Cumulative Charge / Discharge Rate

% ------------------------------------------------------------------------

% Stoic Value at Charge / Discharge Condition
if charge == true                           %↙
-----------------------------------------------
    stoic_n = 0.126;                        % Charge Condition
    stoic_p = 0.936;
else                                        %↙
-----------------------------------------------
    stoic_n = 0.676;                        % Discharge Condition
    stoic_p = 0.442;
end                                         %↙
-----------------------------------------------

% ------------------------------------------------------------------------

% Load Required Parameters
load ( 'temp\grid.mat', 'n_n', 'n_s', 'n_p', 'n_t');
load ( 'temp\material.mat', 'concS_max_n', 'concS_max_p' );
```

```
% -------------------------------------------------------------------------

% Initial Condition for Concentration
conc_o( 1 : n_t ) = 1.2e3;                  % [UNIT: mol / m^3]

concS_o = zeros( 1, n_t );       % Preallocation
for i = 1 : n_n
    concS_o(i) = stoic_n * concS_max_n;         % NEGATIVE
end
for i = n_n + n_s + 1 : n_t
    concS_o(i) = stoic_p * concS_max_p;         % POSITIVE
end

% -------------------------------------------------------------------------

% Initial Guesses - Needed to Calculate Properties at First Time Step

tem( 1 : n_t) = 288;                        % Initial Hardwired Temperature
temFACE( 1 : n_t + 1 ) = 288;               % Hardwired for now
conc( 1 : n_t ) = conc_o( 1 : n_t );        % Initial Concentration
concS( 1 : n_t ) = concS_o( 1 : n_t );      % Initial Solid Face Concentration
concS_surf( 1 : n_t ) = concS( 1 : n_t );   % Iniitla Solid Face Concentration at Surface

phiS = zeros( 1, n_t );                     % Preallocation
eta = zeros( 1, n_t );                      % Preallocation

if charge == true                           %↙
-------------------------------------------------
    phiE( 1 : n_t ) = 1.9;                  % Charge Condition
    for i = 1 : n_n                         % NEGATIVE
        phiS(i) = 2.1;
        phiE(i) = 1.9;
        eta(i) = -0.01;
    end
    for i = n_n + n_s + 1 : n_t             % POSITIVE
        phiS(i) = 5.5;
        phiE(i) = 1.9;
        eta(i) = 0.01;
    end
else                                        %↙
-------------------------------------------------
    phiE( 1 : n_t ) = 2.15;                 % Discharge Condition
    for i = 1 : n_n                         % NEGATIVE
        phiS(i) = 2.2;
        phiE(i) = 2.2;
        eta(i) = 0.01;
    end
    for i = n_n + n_s + 1 : n_t             % POSITIVE
        phiS(i) = 6.0;
        phiE(i) = 2.1;
```

```
        eta(i) = -0.01;
    end                                         %↙
------------------------------------------------
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% TDMA.m

% ************************************************************************

% TDMA - Tridiagonal Matrix Algorithm (Thomas Algorithm)

% Inputs:
%       a - sub-diagonal
%       b - main diagonal
%       c - sup-diagonal
%       d - right part

% Outputs:
%       x - solution

% Parameters:
%       n - number of equations
%       m - temp calulation for matrix elimination

% Requires:
%       a_1 = 0
%       c_n = 0

% ************************************************************************

function x = TDMA( a , b , c , d )

    % number of equations
    n = length( d );

    for i = 2:n
        xmult = a(i-1)/b(i-1);
        b(i) = b(i) - xmult*c(i-1);
        d(i) = d(i) - xmult*d(i-1);
    end

    x(n) = d(n)/b(n);
    for i = n-1:-1:1;
        x(i) = (d(i)-c(i)*x(i+1))/b(i);
    end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% loop_eqpot.m

% ************************************************************************

% Calculate the equilibrium potential ( or open circuit potential)

% Units are all in International System of Units (SI)

% ************************************************************************

eqpot = zeros( 1, n_t );            % Preallocation

% NEGATIVE side
for i = 1 : n_n

    x = ( stoic_n * poroActive_n * concS_max_n * l_n - i_app_cum / FARADAY ) ...
        / ( poroActive_n * concS_max_n * l_n );

    eqpot(i) = 8.00229 + 5.0647 * x - 12.578 * sqrt(x) - 8.6322e-4 / x...
        + 2.1765e-5 * ( x^1.5 ) - 0.46016 * exp ( 15.0 * ( 0.06 - x ) )...
        - 0.55364 * exp ( -2.4326 * ( x - 0.92 ) );

end

% POSITIVE side
for i = n_n + n_s + 1 : n_t

    y = ( stoic_p * poroActive_p * concS_max_p * l_p + i_app_cum / FARADAY ) ...
        / ( poroActive_p * concS_max_p * l_p );

    eqpot(i) = 85.681 * ( y^6 ) - 357.7 * ( y^5 ) + 613.89 * ( y^4 ) ...
        - 555.65 * ( y^3 ) + 281.06 * ( y^2 ) - 76.648 * y ...
        - 0.30987 * exp ( 5.657 * ( y^115 ) ) + 13.1983;

end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% SolidPhasePotential.m

% *************************************************************************

% Solve Solid ( Electronic ) Phase Electric Potential

% Units are all in International System of Units (SI)

% *************************************************************************

% -------------------------------------------------------------------------

% Start of Iteration Loop within Each Time Step - Iteration for phiS
for it_inner = 1 : it_inner_max

    % Some Simplified Coefficients
    jt = zeros( 1, n_t );            % Preallocation
    sc = zeros( 1, n_t );            % Preallocation
    sp = zeros( 1, n_t );            % Preallocation

    for i = 1 : n_n
      i0 = kcon_n * ( conc( i )^alpha_a ) * ( ( concS_max_n - concS_surf( i ) )^alpha_a↙
)...
          * ( concS_surf( i )^alpha_c );
      e_a = alpha_a * FARADAY / ( RU * tem( i ) );
      e_c = alpha_c * FARADAY / ( RU * tem( i ) );
      exp1 = exp( e_a * eta( i ) );
      exp2 = exp( - e_c * eta( i ) );
      expterm = exp1 - exp2;
      deriv = 0;
      jt( i ) = sv( i ) * i0 * expterm;
      deriv = FARADAY * sv( i ) * i0 * ( alpha_a * exp1 + alpha_c * exp2 )...
          / ( RU * tem( i ) );
      sc(i) = - ( jt( i ) - deriv * phiS( i ) );
      sp(i) = -deriv;

    end
    for i = n_n + n_s + 1 : n_t
      i0 = kcon_p * ( conc( i )^alpha_a ) * ( ( concS_max_p - concS_surf( i ) )^alpha_a↙
)...
          * ( concS_surf( i )^alpha_c );
      e_a = alpha_a * FARADAY / ( RU * tem( i ) );
      e_c = alpha_c * FARADAY / ( RU * tem( i ) );
      exp1 = exp( e_a * eta( i ) );
      exp2 = exp( - e_c * eta( i ) );
      expterm = exp1 - exp2;
      jt(i) = sv( i ) * i0 * expterm;
      deriv = FARADAY * sv( i ) * i0 * ( alpha_a * exp1 + alpha_c * exp2 )...
          / ( RU * tem( i ) );
      sc(i) = - ( jt( i ) - deriv * phiS( i ) );
      sp(i) = - deriv;
```

```
end

% ---------------------------------------------------------------------
% Initial TDMA Solver for Electric Potential Equation
res = 0;                    res2 = 0;
                % Reset res & res2
an = zeros( 1, n_n );       bn = zeros( 1, n_n );
cn = zeros( 1, n_n );       dn = zeros( 1, n_n );
                % Preallocation / Reset a, b, c, d for TDMA - NEGATIVE
ap = zeros( 1, n_p );       bp = zeros( 1, n_p );
cp = zeros( 1, n_p );       dp = zeros( 1, n_p );
                % Preallocation / Reset a, b, c, d for TDMA - POSITVE

% ---------------------------------------------------------------------
% Set Up a,b,c,d for TDMA - NEGATIVE

% Left Boundary - NEGATIVE
i = 1;
    bn( i ) = 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    diag = bn( i );
    bn( i ) = bn( i ) - min( 0 , sp( i ) ) * vol( i );
    cn( i ) = - 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    source = - vol( i ) * jt( i ) + i_app;
    dn( i ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i ) + i_app;
    res = ( diag * phiS( i ) + cn( i ) * phiS( i + 1 ) - source )^2;


% Interior Nodes - NEGATIVE
for i = 2 : n_n - 1
    left = 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    right = 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    bn( i ) = left + right;
    diag = bn( i );
    bn( i ) = bn( i ) - min( 0 , sp( i ) ) * vol( i );
    source = - vol( i ) * jt( i );
    dn( i ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i );
    an( i - 1 ) = - left;
    cn( i ) = - right;
    res = res + ( diag * phiS( i ) + cn( i ) * phiS( i + 1 ) + an( i - 1 )...
        * phiS ( i - 1 ) - source )^2;

end

% Right Boundary - NEGATIVE
i = n_n;
  bn( i ) = 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i - 1 ) );
  diag = bn( i );
  bn( i ) = bn( i ) - min( 0 , sp( i ) ) * vol( i );
  an( i - 1 ) = - 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i - 1 ) );
  source = - vol( i ) * jt( i );
  dn( i ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i );
```

```
  res = res + ( diag * phiS( i ) + an( i - 1 ) * phiS( i - 1 )- source )^2;

% Solve System of Equations using TDMA - NEGATIVE

solutionN = TDMA ( an, bn, cn, dn );

for i = 1 : n_n
    phiS( i ) = phiS( i ) + relax * ( solutionN( i ) - phiS( i ) );
    phiS( i ) = min( phiS_max, max( phiS( i ) , phiS_min ) );
end


% ---------------------------------------------------------------------
% Set Up a,b,c,d for TDMA - POSITIVE

% Left Boundary - POSITIVE
il = 1;
i = il + n_n + n_s;
    bp( il ) = 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    diag = bp( il );
    bp( il ) = bp( il) - min( 0 ,sp( i ) ) * vol( i );
    cp( il ) = - 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    source = - vol( i ) * jt( i );
    dp( il ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i );
    res = res + ( diag * phiS( i ) + cp( il ) * phiS( i + 1 ) - source )^2;

% Interior Nodes - POSITIVE
for il = 2 : n_p - 1
    i = il + n_n + n_s;
    left = 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    right = 2 * sigmaEffFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    ap( il - 1 ) = - left;
    bp( il ) = left + right;
    diag = bp( il );
    bp( il ) = bp( il ) - min( 0 , sp( i ) ) * vol( i );
    cp( il ) = - right;
    source = - vol( i ) * jt( i );
    dp( il ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i );
    res = res + ( diag * phiS( i ) + cp( il ) * phiS( i + 1 ) + ap( il - 1 )...
        * phiS( i - 1 ) - source )^2;
end

% Right Boundary - POSITIVE
il = n_p;
i = il + n_n + n_s;
    bp( il ) = 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    diag = bp( il );
    bp( il ) = bp( il ) - min( 0 , sp( i ) ) * vol( i );
    ap( il - 1 ) = - 2 * sigmaEffFACE( i ) / ( dx( i ) + dx( i -1 ) );
    source = - vol( i ) * jt( i ) - i_app;
    dp( il ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiS( i ) -↙
```

```
i_app;
        res = res + ( diag * phiS( i ) + ap( il - 1 ) * phiS( i - 1 ) - source )^2;

        solutionP = TDMA ( ap, bp, cp, dp );

    for il = 1 : n_p;
        i = il + n_n + n_s;
        phiS(i) = phiS( i ) + relax * ( solutionP( il ) - phiS( i ) );
        phiS(i) = min( phiS_max, max( phiS( i ) , phiS_min) );

    end

    %Calculate L2NORM
    res2 = sqrt( max( TINY , res ) );
    if ( it_inner == 1 )
        res_phiS = res2;
    end

    % ---------------------------------------------------------------------
    % Update eta
    eta = zeros( 1, n_t );                  % Preallocation

    for i = 1 : n_t
        eta( i ) = phiS( i ) - phiE( i ) - eqpot( i );
    end

    for i = 1 : n_t
        eta( i ) = max( min( eta_max , eta( i ) ) , eta_min );
    end
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% ElectrolytePhasePotential.m

% *************************************************************************

% Solve Electrolyte Phase Electric Potential

% Units are all in International System of Units (SI)

% *************************************************************************

% -------------------------------------------------------------------------

% Update some of the properties - some time-related properties
loop_update;

% -------------------------------------------------------------------------
% Start of Iteration Loop within Each Time Step - Iteration for phiE
for it_inner = 1 : it_inner_max

    % Some Simplified Coefficients
    jt = zeros( 1, n_t + 1 );            % Preallocation
    sc = zeros( 1, n_t + 1 );            % Preallocation
    sp = zeros( 1, n_t + 1 );            % Preallocation

    for i = 1 : n_n
%         eta(i) = phiS(i) - phiE(i) - eqpot(i);
      i0 = kcon_n * ( conc( i )^alpha_a ) * ( ( concS_max_n - concS_surf( i ) )↙
^alpha_a)...
          * (concS_surf( i )^alpha_c );
      e_a = alpha_a * FARADAY / ( RU * tem( i ) );
      e_c = alpha_c * FARADAY / ( RU * tem( i ) );
      exp1 = exp( e_a * eta( i ) );
      exp2 = exp( - e_c * eta( i ) );
      expterm = exp1 - exp2;
      jt( i ) = sv( i ) * i0 * expterm;
      deriv = -FARADAY * sv( i ) * i0 * ( alpha_a * exp1 + alpha_c * exp2 ) / ( RU * tem(↙
i ) );
      sc( i ) = jt( i ) - deriv * phiE( i );
      sp( i ) = deriv;
    end
    for i = n_n + n_s + 1 : n_t
%         eta(i) = phiS(i) - phiE(i) - eqpot(i)     ;
      i0 = kcon_p * ( conc( i )^alpha_a ) * ( ( concS_max_p - concS_surf( i ) )^alpha_a↙
)...
          * ( concS_surf( i )^alpha_c );
      e_a = alpha_a * FARADAY / ( RU * tem( i ) );
      e_c = alpha_c * FARADAY / ( RU * tem( i ) );
      exp1 = exp( e_a * eta( i ) );
      exp2 = exp( - e_c * eta( i ) );
      expterm = exp1 - exp2;
      jt( i ) = sv( i ) * i0 * expterm;
```

```
      deriv = - FARADAY * sv( i ) * i0 * ( alpha_a * exp1 + alpha_c * exp2 ) / ( RU * tem↙
( i ) );
      sc( i ) = jt( i ) - deriv * phiE( i );
      sp( i ) = deriv;

    end

    % ---------------------------------------------------------------------
    % Initial TDMA Solver for Electric Potential Equation
    res = 0;                    res2 = 0;
                  % Reset res & res2
    a = zeros( 1, n_t );        b = zeros( 1, n_t );
    c = zeros( 1, n_t );        d = zeros( 1, n_t );
                  % Preallocation / Reset a, b, c, d for TDMA - NEGATIVE

    % ---------------------------------------------------------------------
    % Set Up a,b,c,d for TDMA

    % Left Boundary
    i = 1;
        rf = 2 * kapdFACE( i + 1 ) * ( log ( conc( i + 1 ) ) - log ( conc( i ) ) )...
            / ( dx( i ) + dx( i + 1 ) );
        b( i ) = 2 * kapFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
        diag = b( i );
        b( i ) = b( i ) - min( 0 , sp( i ) ) * vol( i );
        c( i ) = - 2 * kapFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
        source = vol( i ) * jt( i ) + rf;
        d( i ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiE( i ) + rf;
        res = ( diag * phiE( i ) + c( i ) * phiE( i + 1 )- source )^2;

    % Interior Nodes
    for i = 2 : n_t - 1
        left = 2 * kapFACE( i ) / ( dx( i ) + dx( i - 1 ) );
        right = 2 * kapFACE( i + 1 ) / ( dx( i ) + dx( i +1 ) );
        rf = 2 * kapdFACE( i + 1 ) * ( log ( conc( i + 1 ) ) - log ( conc( i ) ) )...
            / ( dx( i ) + dx( i + 1 ) );
        lf = 2 * kapdFACE( i ) * ( log ( conc( i ) ) - log ( conc( i - 1 ) ) ) ...
            / ( dx( i ) + dx( i - 1 ) );
        a( i - 1 ) = - left;
        b( i ) = left + right;
        diag = b( i );
        b( i ) = b( i ) - min( 0 ,sp( i ) ) * vol ( i );
        c( i ) = - right;
        source = vol( i ) * jt( i ) + ( rf - lf );
        d( i ) = vol( i ) * sc( i ) + max( 0 ,sp( i ) ) * vol( i ) * phiE( i ) + ( rf -↙
lf );
        res = res + ( diag * phiE( i ) + c( i ) * phiE( i + 1 )...
            + a( i - 1 ) * phiE( i - 1 ) - source )^2;
    end

    % Right Boundary
```

```
    i = n_t;
        lf = 2 * kapdFACE( i ) * ( log ( conc( i ) ) - log ( conc( i - 1 ) ) )...
            / ( dx( i ) + dx( i - 1 ) );
        a( i - 1 ) = - 2 * kapFACE( i ) / ( dx( i ) + dx( i - 1 ) );
        b( i ) = 2 * kapFACE( i ) / ( dx( i ) + dx( i - 1 ) );
        diag = b( i );
        b( i ) = b( i ) - min( 0 , sp( i ) ) * vol( i );

        source = vol( i ) * jt( i ) - lf;
        d( i ) = vol( i ) * sc( i ) + max( 0 , sp( i ) ) * vol( i ) * phiE( i ) - lf;
        res = res + ( diag * phiE( i ) + a( i - 1 ) * phiE( i - 1 ) - source)^2;

    % Solve System of Equations using TDMA
    solution = zeros( 1, n_t );             % Preallocation
    solution = TDMA ( a, b, c, d );

    for i = 1 : n_t
        phiE( i ) = phiE( i ) + relax * ( solution( i ) - phiE( i ) );
        phiE( i ) = min( phiE_max, max( phiE( i ) , phiE_min ) );
    end

    % ---------------------------------------------------------------------
    % Update eta

    eta = zeros( 1, n_t );                  % Preallocation

    for i = 1 : n_t
        eta( i ) = phiS( i ) - phiE( i ) - eqpot( i );
    end

    for i = 1 : n_t
        eta( i ) = max( min( eta_max , eta( i ) ) , eta_min );
    end

    % ---------------------------------------------------------------------
    %Calculate L2NORM
    res2 = sqrt( max( TINY , res ) );
    if ( it_inner == 1 )
        res_phiE = res2;
    end

end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% ElectrolytePhaseConcentration.m

% *************************************************************************

% Solve Electrolyte Phase Concentration

% Units are all in International System of Units (SI)

% *************************************************************************

% -------------------------------------------------------------------------

% Start of Iteration Loop within Each Time Step - Iteration for phiS
for it_inner = 1 : it_inner_max

    % Some Simplified Coefficients
    jt = zeros( 1, n_t + 1 );            % Preallocation
    sc = zeros( 1, n_t + 1 );            % Preallocation
    sp = zeros( 1, n_t + 1 );            % Preallocation

    % -------------------------------------------------------------------


    for i = 1 : n_n
        i0 = kcon_n * ( conc(i)^alpha_a ) * ( ( concS_max_n - concS_surf(i) )^alpha_a)...
            * ( concS_surf( i )^alpha_c );
        e_a = alpha_a * FARADAY / ( RU * tem( i ) );
        e_c = alpha_c * FARADAY / ( RU * tem( i ) );
        exp1 = exp( e_a * eta( i ) );
        exp2 = exp( - e_c * eta( i ) );
        expterm = exp1 - exp2;
        jt( i ) = sv( i ) * i0 * expterm;
        sc( i ) = jt( i ) - alpha_a * jt( i );
        sp( i ) = alpha_a * jt( i ) / ( conc( i ) + TINY );
    end
    for i = n_n + n_s + 1 : n_t
        i0 = kcon_p * ( conc( i )^alpha_a ) * ( ( concS_max_p - concS_surf(i) )↙
^alpha_a)...
            * ( concS_surf( i )^alpha_c );
        e_a = alpha_a * FARADAY / ( RU * tem( i ) );
        e_c = alpha_c * FARADAY / ( RU * tem( i ) );
        exp1 = exp( e_a * eta( i ) );
        exp2 = exp( - e_c * eta( i ) );
        expterm = exp1 - exp2;
        jt( i ) = sv( i ) * i0 * expterm;
        sc( i ) = jt( i ) - alpha_a * jt( i );
        sp( i ) = alpha_a * jt( i ) / ( conc( i ) + TINY );
    end

    % -------------------------------------------------------------------
    % Initial TDMA Solver for Electric Potential Equation
```

```
res = 0;                    res2 = 0;
                % Reset res & res2
a = zeros( 1, n_t );        b = zeros( 1, n_t );
c = zeros( 1, n_t );        d = zeros( 1, n_t );
                % Preallocation / Reset a, b, c, d for TDMA

% ------------------------------------------------------------------
% Set Up a,b,c,d for TDMA

% Left Boundary
i = 1;
    pvt = porosityNODE( i ) * vol( i ) / dt;
    b( i ) = pvt + 2 * diffuFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    diag = b( i );
    b( i ) = b( i ) - min( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) / FARADAY;
    c( i ) = - 2 * diffuFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    source = ( 1 - tPlus ) * vol( i ) * jt( i ) / FARADAY + pvt * conc_o( i );
    d( i ) = ( 1 - tPlus ) * vol( i ) * sc( i ) / FARADAY + pvt * conc_o( i )...
        + max( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) * conc( i ) / FARADAY;
    res = ( diag * conc( i ) + c( i ) * conc( i + 1 ) - source )^2;

% Interior Nodes
for i = 2 : n_t - 1
    left = 2 * diffuFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    right = 2 * diffuFACE( i + 1 ) / ( dx( i ) + dx( i + 1 ) );
    pvt = porosityNODE( i ) * vol( i ) / dt;

    a( i - 1 ) = - left;
    b( i ) = pvt + left + right;
    diag = b( i );
    b( i ) = b( i ) - min( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) / FARADAY;
    c( i ) = - right;
    source = ( 1 - tPlus ) * vol( i ) * jt( i ) / FARADAY + pvt * conc_o( i );
    d( i ) = ( 1 - tPlus ) * vol( i ) * sc( i ) / FARADAY + pvt * conc_o( i )...
        + max( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) * conc( i ) / FARADAY;
    res = res + ( diag * conc( i ) + c( i ) * conc( i + 1 ) + a( i - 1 )...
        * conc( i - 1 ) - source )^2;
end

% Right Boundary
i = n_t;
    pvt = porosityNODE( i ) * vol( i ) / dt;
    a( i - 1 ) = - 2 * diffuFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    b( i ) = pvt + 2 * diffuFACE( i ) / ( dx( i ) + dx( i - 1 ) );
    diag = b( i );
    b( i ) = b( i ) - min( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) / FARADAY;
    source = ( 1 - tPlus ) * vol( i ) * jt( i ) / FARADAY + pvt * conc_o( i );
    d( i ) = ( 1 - tPlus ) * vol( i ) * sc( i ) / FARADAY + pvt * conc_o( i )...
        + max( 0 , sp( i ) ) * ( 1 - tPlus ) * vol( i ) * conc( i ) / FARADAY;
    res = res + ( diag * conc( i ) + a( i - 1 ) * conc( i - 1 )- source )^2;
```

```
    % ---------------------------------------------------------------------
    % Solve System of Equations using TDMA
    solution = zeros( 1, n_t + 1 );             % Preallocation
    solution = TDMA ( a, b, c, d );

    for i = 1 : n_t
        conc( i)  = conc( i ) + relax * (solution( i ) - conc( i ) );
        conc( i ) = min( conc_max , max( conc( i ) , conc_min ) );
    end

    % ---------------------------------------------------------------------
    % Calculate L2NORM
    res2 = sqrt( max ( TINY, res ) );
    if( it_inner == 1 )
        res_conc = res2;
    end
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% SolidPhaseConcentration.m

% *************************************************************************

% Solve Li-ion Solid Phase Surface Concentration

% Units are all in International System of Units (SI)

% *************************************************************************

    n_r = 11;
    dr = R_s / ( n_r-1);
    dr2 = dr*dr;
% -------------------------------------------------------------------------
% Start of Iteration Loop within Each Time Step - Iteration for concS
for it_inner = 1 : it_inner_max

    concS_star = concS;

    % Some Simplified Coefficients
    jt = zeros( 1, n_t );           % Preallocation
    sc = zeros( 1, n_t );           % Preallocation
    sp = zeros( 1, n_t );           % Preallocation

    for i = 1 : n_n
        i0 = kcon_n * ( conc( i )^alpha_a ) * ( ( concS_max_n - concS_surf( i ) )^alpha_a↙
)...
            * ( concS_surf( i )^alpha_c );
        e_a = alpha_a * FARADAY / ( RU * tem( i ) );
        e_c = alpha_c * FARADAY / ( RU * tem( i ) );
        exp1 = exp( e_a * eta( i ) );
        exp2 = exp( - e_c * eta( i ) );
        expterm = exp1 - exp2;
        jt( i ) = sv( i ) * i0 * expterm;
        deriv = jt( i ) * ( - alpha_a / ( ( concS_max_n - concS_surf( i ) ) + TINY )...
            + alpha_c / ( concS_surf( i ) + TINY ) );
        sc( i ) = - ( jt( i ) - deriv * concS_surf( i ) );
        sp( i ) = - deriv;
    end
    for i = n_n + n_s + 1 : n_t
        i0 = kcon_p * ( conc( i )^alpha_a ) * ( ( concS_max_p - concS_surf( i ) )^alpha_a↙
)...
            * ( concS_surf( i )^alpha_c );
        e_a = alpha_a * FARADAY / ( RU * tem( i ) );
        e_c = alpha_c * FARADAY / ( RU * tem( i ) );
        exp1 = exp( e_a * eta( i ) );
        exp2 = exp( - e_c * eta( i ) );
        expterm = exp1 - exp2;
        jt( i ) = sv( i ) * i0 * expterm;
        deriv = jt( i ) * ( - alpha_a / ( ( concS_max_n - concS_surf( i ) ) + TINY)...
            + alpha_c / ( concS_surf( i ) + TINY ) );
```

```
    sc( i ) = - ( jt( i ) - deriv * concS_surf( i ) );
    sp( i ) = - deriv;
end

% -------------------------------------------------------------------
% Initial TDMA Solver for Electric Potential Equation
res = 0;                    res2 = 0;
                % Reset res & res2

% -------------------------------------------------------------------
% NEGATIVE
for i = 1 : n_n
    cp_o( 1 : n_r ) = concS_o( i );

    a = zeros( 1, n_r );        b = zeros( 1, n_r );
    c = zeros( 1, n_r );        d = zeros( 1, n_r );
                % Preallocation / Reset a, b, c, d for TDMA

    j = 1;
        b(1) = 1 / dt + 2.0 * diffuSolid_n / dr2;
        c(1) = - 2.0 * diffuSolid_n / dr2;
        d(1) = cp_o( 1 ) / dt;

    for j = 2 : n_r - 1
        r0 = j * dr;
        a( j - 1 ) = - ( diffuSolid_n / dr2 - diffuSolid_n / ( r0 * dr ) );
        b( j ) = 1 / dt + 2.0 * diffuSolid_n / dr2;
        c( j ) = - ( diffuSolid_n / dr2 + diffuSolid_n / ( r0 * dr ) );
        d( j ) = cp_o( j ) / dt;
    end

    a( n_r - 1 ) = - 2.0 * diffuSolid_n / dr2;
    b( n_r ) = 1 / dt + 2.0 * diffuSolid_n / dr2 - min( 0 , sp( i ) )...
        * 2.0 * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY);
    d( n_r ) = 2.0 * sc( i ) * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY )...
        + cp_o( n_r ) / dt + max( 0 , sp( i ) ) * concS_surf( i )...
        * 2.0 * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY );

    solution1 = TDMA( a, b, c, d);

    % Calculate Average Concentration in Particles
    sumc = 0;
    sumr = 0;
    for j = 2 : n_r - 1
        r0 = j *dr;
        sumc = sumc + solution1(j) * r0^2 * dr;
        sumr = sumr + r0^2 *dr;

    end
    sumc = sumc + solution1( j ) * R_s*R_s * dr / 2;
    sumr = sumr + R_s^2 * dr / 2;
```

```
        cavg = sumc / sumr;
        concS( i ) = concS( i ) + relax * (cavg - concS( i ) );
        concS_surf( i ) = concS_surf( i ) + relax * (solution1( n_r ) - concS_surf( i ) );
        concS( i ) = min ( max ( conc_min , concS( i ) ) , concS_max_n );
        concS_surf( i ) = min ( max ( conc_min , concS_surf( i ) ) , concS_max_n );

        res = res + ( concS_star( i ) - concS( i ) )^2;
    end

    % SEPARATER
    for i = n_n + 1 : n_n + n_s
        concS( i ) = 0;
    end

    % POSITIVE
    for i = n_n + n_s + 1 : n_t
        cp_o( 1 : n_r ) = concS_o( i );

        a = zeros( 1, n_r );        b = zeros( 1, n_r );
        c = zeros( 1, n_r );        d = zeros( 1, n_r );
                    % Preallocation / Reset a, b, c, d for TDMA

        j = 1;
            b( 1 ) = 1 / dt + 2.0 * diffuSolid_p / dr2;
            c( 1 ) = - 2.0 * diffuSolid_p / dr2;
            d( 1 ) = cp_o( 1 ) / dt;

        for j = 2 : n_r-1
            r0 = j * dr;
            b( j ) = 1 / dt + 2.0 * diffuSolid_p / dr2;
            c( j ) = - ( diffuSolid_p / dr2 + diffuSolid_p / ( r0 * dr ) );
            a( j - 1 ) = - ( diffuSolid_p / dr2 - diffuSolid_p / ( r0 * dr ) );
            d( j ) = cp_o( j ) / dt;
        end

        b( n_r ) = 1 / dt + 2.0 * diffuSolid_p / dr2 - min ( 0 , sp( i ) )...
            * 2.0 * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY );
        a( n_r - 1 ) = - 2.0 * diffuSolid_p / dr2;
        d( n_r ) = 2.0 * sc( i ) * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY )...
            + cp_o( n_r ) / dt + max( 0 , sp( i ) ) * concS_surf( i ) * 2.0...
            * ( 1 / dr + 1 / R_s ) / ( sv( i ) * FARADAY);

        solution2 = TDMA ( a, b, c, d );

        % Calculate average concentration in particle\
        sumc = 0;
        sumr = 0;
        for j = 2 : n_r - 1
            r0 = j * dr;
```

```
            sumc = sumc + solution2( j ) * r0^2 * dr;
            sumr = sumr + r0^2 * dr;
        end

        sumc = sumc + solution2( j ) * R_s*R_s * dr / 2;
        sumr = sumr + R_s^2 * dr / 2;

        cavg = sumc / sumr;
        concS( i ) = concS(i) + relax * ( cavg - concS( i ) );
        concS_surf( i ) = concS_surf( i ) + relax * (solution2( n_r ) - concS_surf( i )↙
);
        concS( i ) = min ( max ( conc_min , concS( i ) ) , concS_max_n );
        concS_surf( i ) = min ( max ( conc_min , concS_surf( i ) ) , concS_max_p );
        res = res + ( concS_star( i ) - concS( i ))^2;
    end

    % Calculate L2NORM
    res2 = sqrt( max ( TINY , res ) );
    if ( it_inner == 1)
        res_concS = res2;
    end
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% loop_check.m

% ************************************************************************

% Check the convergence of the iteration solution
% If converges, go to next time step
% If not, return to iteration until converges

% Units are all International System of Units (SI)

% ************************************************************************

converged = true;
res_max( 1 ) = max(res_max(1),res_conc);
res_max( 2 ) = max(res_max(2),res_concS);
res_max( 3 ) = max(res_max(3),res_phiS);
res_max( 4 ) = max(res_max(4),res_phiE);
normres( 1 ) = res_conc/res_max(1);
normres( 2 ) = res_concS/res_max(2);
normres( 3 ) = res_phiS/res_max(3);
normres( 4 ) = res_phiE/res_max(4);
for i = 1:4
    if (normres(i) > tol_outer)
        converged = false;
        break;
    end
end

if(converged)
    fprintf('time=%4d sec    iter=%4d        normres=%.4e\n',it, iter, max(normres))
end

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% loop_update.m

% ************************************************************************

% Update some of the properties - some time-related properties

% Units are all International System of Units (SI)

% ************************************************************************

% ------------------------------------------------------------------------

% Ionic and diffusive conductivity of each cell
kapNODE = zeros( 1, n_t );                          % Preallocation
kapdNODE = zeros( 1, n_t );                         % Preallocation

for i = 1 : n_t
    ce = conc(i) * 1.0e-6;       % mol / cm^3
    kap_fs = 15.8 * ce * exp( 0.85 * ( ( 1000.0 * ce )^1.4 ) );     % S / cm
    kap_fs = kap_fs * 100.0;        % S / m
    kapNODE(i) = kap_fs * ( porosityNODE(i)^1.5 );
    kapdNODE(i) = 2 * RU * tem(i) * (tPlus - 1 ) * kapNODE(i) / FARADAY;
end

% ------------------------------------------------------------------------
% Transport properties at Faces
kapFACE = zeros( 1, n_t + 1 );                % Preallocation
kapdFACE = zeros( 1, n_t + 1 );               % Preallocation


% Left Boundary
kapFACE(1) = kapNODE(1);

% NEGATIVE
for i = 2 : n_n
    kapFACE(i) = ( kapNODE( i - 1 ) + kapNODE( i ) ) / 2;
end

% NEGATIVE and SEPARATER Boundary
wt = ( 1 / dx( n_n ) ) / ( 1 / dx( n_n ) + 1 / dx( n_n + 1 ) );
avg_k = wt * kapNODE( n_n ) + ( 1 - wt ) * kapNODE( n_n + 1 );
kapFACE( n_n + 1 ) = kapNODE( n_n ) * kapNODE( n_n + 1 ) / avg_k;

% SEPARATER
for i = n_n + 2 : n_n + n_s
    kapFACE(i) = ( kapNODE( i - 1 ) + kapNODE( i ) ) / 2;
end

% POSITIVE and SEPARATER Boundary
wt = ( 1 / dx( n_n + n_s ) ) / ( 1 / dx( n_n + n_s ) + 1 / dx( n_n + n_s + 1 ) );
avg_k = wt * kapNODE( n_n + n_s ) + ( 1 - wt ) * kapNODE( n_n + n_s + 1 );
```

```
kapFACE( n_n + n_s + 1 ) = kapNODE( n_n + n_s ) * kapNODE( n_n + n_s + 1 ) / avg_k;

% POSITIVE
for i = n_n + n_s + 2 : n_t
    kapFACE(i) = ( kapNODE( i - 1 ) + kapNODE( i ) ) / 2;
end

% Right Boundary
kapFACE( n_t + 1 ) = kapNODE( n_t );

% -------------------------------------------------------------------------

for i = 1 : n_t + 1
    kapdFACE( i ) = 2 * RU * temFACE(i) * ( tPlus - 1 ) * kapFACE(i) / FARADAY;
end

% -------------------------------------------------------------------------
% Concentration at Faces
concFACE(1) = conc(1);
for i = 2 : n_t-1
    concFACE(i) = 1/2 * ( conc(i) + conc(i-1) );
end
concFACE(n_t) = conc(n_t);

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% time_march.m

% ************************************************************************

% Time march

% Units are all International System of Units (SI)

% ************************************************************************

% Reset maximum residual
res_max(:) = TINY;

% Time advance all variables
conc_o = conc;                          % Electrolyte Phase Concentration
concS_o = concS;                        % Solid Phase Concentration
if(hybrid_cycle)
    i_app = app_curr(it);
end
i_app_cum = i_app_cum + i_app * dt;     % Integral i*dt

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```

```
% post.m

% ************************************************************************

% Post Process

% Units are all International System of Units (SI)

% ************************************************************************

% Calculate battery voltage

v_left = phiS( 1 ) + i_app * dx( 1 ) * 0.5 / ( sigmaFACE( 1 ) );
v_right = phiS( n_t ) - i_app * dx( n_t ) * 0.5 / ( sigmaFACE( n_t + 1 ) );
cell_voltage = v_right - v_left - resistance * i_app;


if( hybrid_cycle )
    display('hybrid cycle')
%     fprintf( it * dt, app_curr(it), cell_voltage );
else
    if( charge == false )
        dod = i_app * it * dt / ( capacity * 3600.0 );      % Normalized depth of↙
discharge
        fprintf( 'dod =%3.4f\t\tCell Voltage =%3.4f\n' ,  dod,  cell_voltage );
    else
        soc = - i_app * it * dt / ( capacity * 3600.0);     % Normalized state of charge
        fprintf( 'soc =%3.4f\t\tCell Voltage =%3.4f\n' ,  soc,  cell_voltage );
    end
end

fprintf( '\t\t\t\tOpen Circuit Voltage =%3.4f\n\n' ,  eqpot( n_t ) - eqpot( 1 ) );

% DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE***DONE
```